

1 Dual-based methods for solving infinite-horizon nonstationary
2 deterministic dynamic programs

3 Christopher Thomas Ryan* Robert L Smith†

4 February 11, 2020

5 **Abstract**

6 We develop novel dual-ascent and primal-dual methods to solve infinite-horizon nonstationary
7 deterministic dynamic programs. These methods are finitely implementable and converge in
8 value to optimality. Moreover, the dual-ascent method produces a sequence of improving dual
9 solutions that pointwise converge to an optimal dual solution, while the primal-dual algorithm
10 provides a sequence of primal basic feasible solutions with value error bounds from optimality
11 that converge to zero. Our dual-based methods work on a more general class of infinite network
12 flow problems that include the shortest-path formulation of dynamic programs as a special case.
13 To our knowledge, these are the first dual-based methods proposed in the literature to solve
14 infinite-horizon nonstationary deterministic dynamic programs.

15 **1 Introduction**

16 Industry is often accused of being short-sighted in its planning, obsessed with next quarter's pro-
17 jected profit margin. Many planning tools themselves exacerbate this focus by only considering
18 decisions over a fixed finite horizon. Typically, tools that incorporate an infinite horizon make sta-
19 tionarity assumptions that restrict the future to be exactly the world we confront today. Attempts
20 to tackle infinite-horizon and nonstationary environments have been few in number. Examples
21 of this nonstationary setting in practice include the sizing and timing of capacity expansions [4],
22 planning for production scheduling [35], the replacement and acquisition of new equipment [5],
23 dynamic traffic assignment [21], among others. These problems can be modeled as infinite-horizon
24 nonstationary dynamic programs (or simply DPs), the context of this paper.

25 How do we deal with a problem that is nonstationary and non-ending? Tackling this challenge
26 invites many unsavory characters into our midst: an infinite amount of data, pathologies of infinite-
27 dimensional spaces, challenges in expressing solutions finitely, etc. Past attempts to tackle this

*Booth School of Business, University of Chicago, E-mail: chris.ryan@chicagobooth.edu

†Industrial and Operations Engineering, University of Michigan, E-mail: rlsmith@umich.edu

28 problem have followed one of two approaches. The first approach, called the planning horizon
29 approach, is to (if possible) find a finite horizon T sufficiently distant to make the associated errors
30 from ignoring times beyond T negligible [3, 8, 9, 25]. A key insight here is that although end-of-
31 horizon effect distort decisions close to T , the near term (in particular, the first decision) is less
32 affected. This decoupling of the present and future decisions can be achieved by discounting [3],
33 uncertainty [1, 19], or through tie-breaking selection [31, 33]. Luckily, the current decision is the
34 only one the decision-maker must commit to at the time of implementation. If calculations are
35 efficient, a rolling horizon approach is feasible.

36 The second approach, typically called the strategy horizon approach, searches for optimal so-
37 lutions within the space of infinite-horizon decision sequences, or strategies. Most success in this
38 direction is for stationary problems, where the model is simplified to allow for analytic, and hence
39 finite, solutions. An example is solving infinite-horizon homogeneous Markov decision processes
40 that can restrict search within stationary strategies [10, 27, 29]. The difficulty here is that the
41 necessary model simplifications may exclude many realistic problems.

42 Since nonstationary data is sometimes unavoidable, other strategy-horizon approaches have
43 been developed. Most notable are solving DPs as infinite-dimensional linear programs. Simplex
44 methods and duality theory are then leveraged to establish properties of optimal strategies without
45 resort to finite truncations of the horizon. References [16] and [24] develop simplex methods to
46 solve nonhomogeneous versions of Markov decision processes (MDPs) based on their formulation
47 as countably-infinite linear programs (CILPs). Several authors have also adapted Anderson and
48 Nash’s [2] general framework for infinite-dimensional linear programs for Markov decision processes
49 to solve *uncountable* state space stationary Markov and semi-Markov decision problems using linear
50 programming formulations of Bellman equations. See for instance, [14, 18, 22, 23].

51 The simplex methods in [16] and [24] rely on the condition that all extreme points are *nonde-*
52 *generate*, a crucial property for showing asymptotic convergence. By contrast, a deterministic DP
53 corresponds to a CILP that is highly *degenerate*. To the authors’ knowledge, only three papers
54 study deterministic DP via a strategy horizon approach using CILP formulations. Ghatge, Sharma
55 and Smith [17] develop a simplex-like method they call the *shadow simplex method* for a more
56 general class of CILPs that have deterministic DP as a special case (see their Section 4). Sharkey
57 and Romeijn [34] propose a simplex method for capacitated infinite network flow problems that can
58 model deterministic dynamic programming as a shortest path problem (following the construction
59 we set out in [Section 2](#) below). Ryan, Smith, and Epelman [30] propose a model for uncapacitated
60 network flow problems and model deterministic DP as an all-to-infinity shortest-path problem in
61 their Section 5.2.

62 All three approaches in these papers have their limitations. Each can be shown to converge in
63 optimal value, meaning iterates of their respective simplex methods converge in value to optimality.
64 However, each fails to guarantee solution convergence, where successive policy iterates converge to

65 an optimal solution unless there is a unique optimal solution to the problem. Moreover, the methods
66 of [34] and [30] may not even be finitely-implementable, meaning each iteration of the algorithm
67 may run in infinite time in the worst case.

68 In this paper, we build on the “infinite network” view of dynamic programming, but instead of
69 constructing a primal simplex method, as in [34] and [30], we generalize dual-ascent and primal-dual
70 methods for finite network flow problems (see, for instance, Chapter 9 of [7]) to the infinite setting.
71 This setting captures deterministic DP as a special case. Unlike the primal methods discussed in
72 the previous paragraph, we show that the dual-ascent method is finitely-implementable and has
73 guaranteed pointwise solution convergence. The primal-dual method adapts the dual-ascent method
74 to concurrently generate primal feasible solutions in the form of spanning trees. An optimality error
75 bound on the gap between objective value of these primal feasible solutions and the optimal value
76 is provided, based on the current dual solution. This optimality guarantee is refined as the primal-
77 dual method proceeds and the gap disappears in the limit. Both the dual-ascent method and
78 primal-dual method leverage the special structure of the infinite graphs we study and cannot easily
79 be seen as a straightforward extension of the corresponding methods in the finite-network case.

80 To our knowledge, this paper is the first to develop dual-based algorithms for infinite network
81 flow problems. There has been a recent surge of work on infinite network flow problems and their
82 generalizations (see [26] and [15], in addition to [30] and [34] discussed above) but these works either
83 focus on weak and strong duality properties or primal simplex methods, not dual-based methods.
84 Moreover, in this paper we establish strong duality using dual-based arguments, a departure from
85 previous approaches.

86 Our work also relates to a flurry of recent progress on approximate solutions to infinite-
87 dimensional LPs for solving stationary stochastic DPs (see, for instance, [11–13, 32]). These papers
88 finds ways to approximate infinite LPs by finite-dimensional optimization problems under certain
89 assumptions (compact state and action spaces, or sufficient continuity in the data) and provide
90 explicit error bounds on the quality of their approximations.

91 The algorithms that we describe in this paper are exact algorithms, but since the full run times
92 of our methods are infinite, finite termination provides approximately optimal solutions. Like
93 the existing literature, we show that if our algorithms run long enough these finite approximations
94 converge in value. However, unlike some recent papers (for instance, those described in the previous
95 paragraph) we are *not* able to provide explicit error bounds for our approximations. To the best
96 of our knowledge, there are no explicit error bounds for the discrete, non-compact case for infinite-
97 horizon dynamic programming in the literature.

98 It is important to note that the approximation algorithms developed in the literature that *do*
99 give explicit error bounds do *not* apply to our setting. This is because our effective state and
100 action spaces are not compact while those algorithms, including [12, 13, 32], require compactness
101 or strong continuity properties. For example, consider, the recent paper [13]. Assumption 2.1(i)

102 requires that the state and action spaces lie in the unit hypercube of the appropriate dimensional
 103 space (or at least can be transformed into such a hypercube). This requires that the state-action
 104 set is compact, as stated in the paragraph above Theorem 2.2 in [13]. In particular, one can note
 105 that the definition of the transition kernel in that paper $Q(\cdot|s, a)$ is not indexed by time. To put a
 106 nonstationary problem into this setting would include time as a component of the state and thus
 107 make the state space noncompact. (Details on such a formulation are included in Section 2.1.)
 108 Accordingly, these results do not apply to our (noncompact) setting.

109 To further underscore the difference between the compact and noncompact settings, another
 110 reference [32] shows asymptotic convergence properties of algorithms in the noncompact case (as
 111 we do in our paper) but must restrict to the compact case when deriving explicit error bounds.
 112 Other references, such as [12], allow noncompact state spaces in their derivation of explicit error
 113 bounds, but must impose strong continuity conditions on the transition kernel. These conditions
 114 fail in the discrete time setting when time is included in the state.

115 This paper is organized as follows. In Section 2, we introduce the dynamic program we study
 116 and formulate it as an infinite network flow problem. In Section 3 the more general class of pure-
 117 supply network flow problems is introduced. Section 4 describes our dual ascent method. Section 5
 118 describes our primal-dual method which builds on the primal method of [30] and the dual ascent
 119 method of the previous section. Section 6 concludes.

120 2 Infinite-horizon nonstationary dynamic programming

121 We consider a nonstationary infinite-horizon deterministic dynamic programming problem (or sim-
 122 ply a DP) with finite per period state and action spaces, defined as follows. A system evolves over
 123 time periods $t = 0, 1, 2, \dots$ in one of finitely many states $s_t \in S_t$ in each period. The starting state
 124 s_0 is arbitrarily chosen from the set S_0 . An action a_t is chosen from a finite and nonempty set $A_{s,t}$
 125 that depends on each state $s \in S_t$ and time t . The system transitions to a new state according to
 126 the law $s' = \tau_t(s_t, a_t) \in S_{t+1}$, yielding a (nonnegative and undiscounted) immediate cost $c_t(s_t, a_t)$.
 127 If there exists a state $s' \in S_{t+1}$ that is not reachable from any state in S_t (that is, if there does
 128 not exist an $s \in S_t$ and $a \in A_{s,t}$ with $s' = \tau_t(s, a)$) then we remove state s' from S_{t+1} since it is
 129 redundant.

130 There is a time-discounting factor $\delta \in (0, 1)$ associated with costs. The discounted cost at time
 131 t is $\delta^t c_t(s, a)$. A policy $d = \{d(\cdot, t) : t = 0, 1, 2, \dots\}$ prescribes an action $d(s, t) \in A_{s,t}$ for every
 132 $s \in S_t$ and every t . Let D denote the set of all policies from starting state s_0 . The strategy $\pi(d)$
 133 corresponding to policy d is the sequence of actions $\pi_t(d)$ out of initial state s_0 provided by policy d
 134 where $\pi_t(d) = d(s_t(d), t)$ and $s_{t+1}(d) = \tau_t(s_t(d), d(s_t(d), t))$ for $t = 0, 1, 2, \dots$ and $s_0(d) = s_0$. Then
 135 $V(\pi(d), s_0) := \sum_{t=0}^{\infty} \delta^t c_t(s_t(d), \pi_t(d))$ is the cost of policy d (and its corresponding strategy $\pi(d)$)
 136 with starting state s_0 .

137 We assume some additional structure as follows:

- 138 (S1) immediate costs are uniformly bounded by $\bar{c} < \infty$; that is, $0 \leq c_t(s, a) \leq \bar{c}$ for all s, a .
 139 (S2) the cardinality of the set of states S_t at time t is uniformly bounded by a constant G ;
 140 that is, $\#(S_t) \leq G$ for all $t = 0, 1, \dots$.

141 The focus of this paper is how to determine a policy that minimizes total discounted cost. That
 142 is, we solve

$$143 \quad V^* := \min_{d \in D} V(\pi(d), s_0), \quad (1)$$

144 where s_0 is a fixed starting state. We consider a non-stationary version of the problem, where state
 145 sets S_t , available actions $A_{s,t}$, immediate costs c_t , and transition laws τ_t can all depend on t . Our
 146 approach to solving this problem is to formulate it as a minimum cost network flow problem on a
 147 network with countably-many nodes (see [Section 3](#) below).

148 We define the network $N = (\mathcal{N}, \mathcal{A}, b, c)$ as follows. The set \mathcal{N} consists of nodes for each state-
 149 time pair (s, t) where $t = 0, 1, 2, \dots$ and $s \in S_t$. The set \mathcal{A} consists of arcs $((s, t), (s', t+1))$ between
 150 pairs of nodes for which there exists $a \in A_{s,t}$ such that $\tau_t(s, a) = s'$. Arcs are uncapacitated. Node
 151 $(s_0, 0)$ is endowed with a unit supply of 1 and all other nodes have a supply of 0, which defines the
 152 vector of supplies b . Each arc has an associated cost $c_{((s,t),(s',t+1))} = \delta^t c_t(s, a)$, which defines arc
 153 costs c . Note that this assumes that there is only a single cost (namely $c_t(s, a)$) for each action a in
 154 $A_{s,t}$ that transitions the problem to state s' at time t , otherwise $c_{((s,t),(s',t+1))}$ is multiply defined.
 155 This is without loss of generality. Only the lowest costs actions that transition from s to s' at time
 156 t are considered and so we may assume that only one cost $c_t(s, a)$ persists. By the same reasoning,
 157 we may assume that only a single *action* exists that transitions from state s to s' at time t .

158 **Remark 1.** In the stochastic DP setting, we cannot remove “redundant” actions as we have done
 159 here simply based on comparing their costs. This is because two actions may give rise to the same
 160 cost, but not to the same transition probabilities. Only when two actions have the same costs and
 161 state transitions probabilities can one of them be considered “redundant”.

162 Using this network we may define a min-cost flow problem as follows. For brevity, for every node
 163 $(s, t) \in \mathcal{N}$, let $O(s, t)$ denote the set $\{(s', t+1) : \tau_t(s, a) = s' \text{ for some } a \in A_{s,t}\}$ of nodes incident
 164 to outgoing arcs of (s, t) and $I(s, t)$ denote the set $\{(s', t-1) : \tau_{t-1}(s', a) = s \text{ for some } a \in A_{s',t-1}\}$
 165 of nodes incident to incoming arcs into (s, t) . Note that $I(s_0, 0) = \emptyset$ since the system starts at time
 166 0 in state s_0 .

167 The network formulation of (1), given the starting state s_0 , is:

$$168 \quad Z^* := \min_x \sum_{((s,t),(s',t+1)) \in \mathcal{A}} \delta^t c_{(s,t)(s',t+1)} x_{(s,t)(s',t+1)} \quad (2a)$$

$$169 \quad \text{subject to} \quad \sum_{(s',1) \in O(s_0,0)} x_{(s_0,0)(s',1)} = 1 \quad (2b)$$

$$170 \quad \sum_{(s',t+1) \in O(s,t)} x_{(s,t)(s',t+1)} - \sum_{(s',t-1) \in I(s,t)} x_{(s',t-1)(s,t)} = 0 \text{ for } (s, t) \in \mathcal{N} \quad (2c)$$

171
172

$$x_{(s,t)(s',t+1)} \geq 0 \text{ for } ((s,t), (s',t+1)) \in \mathcal{A}, \quad (2d)$$

173 where $x_{((s,t)(s',t+1))}$ is interpreted as the flow along arc $((s,t), (s',t+1))$. Let

174

$$Z(x) := \sum_{((s,t),(s',t+1)) \in \mathcal{A}} \delta^t c_{(s,t)(s',t+1)} x_{(s,t)(s',t+1)}.$$

175

The next result relates problems (1) and (2). Details of the proof are contained in the appendix.

176

The argument there follows familiar reasoning from the finite-dimensional case, but must take

177

special care of some topological issues that arise in the infinite-dimensional setting.

178

Proposition 1. Problem (1) and (2) are equivalent in the following sense:

179

(i) every optimal policy d^* can be used to construct an optimal flow x^* where $V(\pi(d^*), s_0) = Z(x^*)$.

180

181

(ii) there exists an optimal flow x^* that can be used to construct an optimal policy d^* where $V(\pi(d^*), s_0) = Z(x^*)$.

182

183

A complete and careful proof of this proposition is in the appendix. Here, we provide the reader with an idea of how the proof works. Note that every feasible flow x to (2) must satisfy $x \in [0, 1]^{\mathcal{A}}$.

184

This is an immediate consequence of constraints (3b)–(6c). Consequently, every integral solution

185

satisfies $x \in \{0, 1\}^{\mathcal{A}}$ and so (again by (6b)) can be associated with a unique path from node s_0

186

to infinity. This path corresponds to a policy that is feasible to (1). Moreover, it can be shown

187

that every optimal solution to (2) is integral (see Lemma 14 in the appendix for details). Thus,

188

the original formulation (1) and the network flow formulation (2) are equivalent in the following

189

sense: every integral solution to (2) corresponds to a feasible solution to (1) (and vice versa) and

190

they have the same objective value. Thus, since optimal solutions to (2) are integral, the result in

191

Proposition 1 holds.

192

193

We also note that the equivalence expressed in Proposition 1 is quite strong. There is a one-to-

194

one correspondence between the optimal solutions of these problems. Other papers in the literature

195

have discussed weaker forms of equivalence (see, for instance, Theorem 1 in [20]).

196

The following provides an explicit example of the equivalence in Proposition 1.

197

Example 1. Consider the following DP. The set of actions S_t is not changing with t and consists

198

of two states; that is, $S_t = S = \{1, 2\}$ for all t . The decision-maker has action set $A_{1,t} = \{1, 2\}$ in

199

state 1 (for all t) and singleton action set $A_{2,t} = \{1\}$ in state 2 (for all t). The transitions are as

200

follows:

201

$$\tau_t(s, a) = \begin{cases} 1 & \text{if } s = 1 \text{ and } a = 1 \\ 2 & \text{if } s = 2 \text{ or } a = 2 \end{cases}$$

202

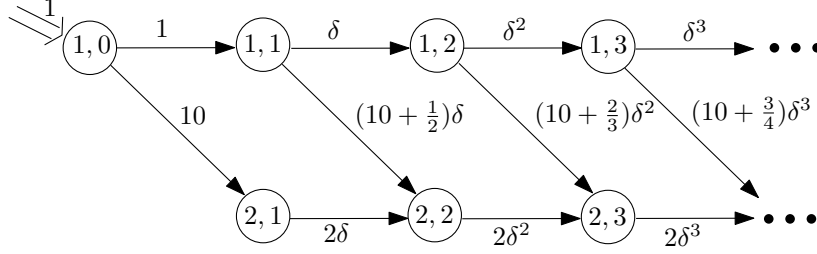


Figure 1: A simple example of the network representation of a nonstationary DP. We follow the conventions of [7] for drawing networks. Arc costs are found on top of arcs. Wide arrows (\Rightarrow) represents node supplies.

for all t and the costs are

$$c_t(s, a) = \begin{cases} 2 & \text{if } s = 2 \\ 1 & \text{if } s = 1, a = 1 \\ 10 + \frac{t}{t+1} & \text{if } s = 1, a = 2 \end{cases}$$

for all t . Note that the costs are nonstationary. State 1 is the starting state s_0 . The network representation is found in Figure 1. The network formulation (2) in this instance is:

$$\min_x \sum_{t=0}^{\infty} \delta^t x_{(1,t)(1,t+1)} + \sum_{t=1}^{\infty} 2\delta^t x_{(2,t)(2,t+1)} + \sum_{t=0}^{\infty} (10 + \frac{t}{t+1}) \delta^t x_{(1,t)(2,t+1)} \quad (3a)$$

$$\text{subject to } x_{(1,1)(1,2)} + x_{(1,1)(2,2)} = 1 \quad (3b)$$

$$x_{(1,t)(1,t+1)} - x_{(1,t-1)(1,t)} = 0 \quad (3c)$$

$$x_{(2,t)(2,t+1)} - (x_{(2,t-1)(2,t)} + x_{(1,t-1)(2,t)}) = 0 \quad (3d)$$

$$x_{(s,t)(s',t+1)} \geq 0 \text{ for } s = 1, 2 \text{ and } t = 0, 1, \dots \quad (3e)$$

2.1 Connection to infinite-horizon stationary stochastic dynamic programming

The network formulation in (2) for nonstationary deterministic DP may be familiar to those familiar with the LP formulation of infinite-horizon stationary stochastic dynamic programming, discussed in detail in numerous references (see, for instance, Chapter 6 of [27]). We describe this formulation in detail here to draw comparisons with our formulation of the deterministic setting.

In the stationary stochastic (Markov) DP setting, we have a set of states Σ and an action set \mathcal{A}_σ for each $\sigma \in S$. Neither Σ nor \mathcal{A}_σ change over time. The problem is stochastic, reflected by a transition probability $p(\sigma'|\sigma, \alpha)$ of transitioning to state σ' when starting in state σ and taking

222 action α . Puterman [27] (among many others) gives an LP formulation of this problem:¹

$$223 \quad \min_x \sum_{\sigma \in \Sigma} \sum_{\alpha \in \mathcal{A}_\sigma} c(\sigma, \alpha) x(\sigma, \alpha) \quad (4a)$$

$$224 \quad \text{subject to } \sum_{\alpha \in \mathcal{A}_j} x(j, \alpha) - \delta \sum_{\sigma \in \Sigma} \sum_{\alpha \in \mathcal{A}_\sigma} p(j|\sigma, \alpha) x(\sigma, \alpha) = \alpha(j) \text{ for all } j \in \Sigma \quad (4b)$$

$$225 \quad x(\sigma, \alpha) \geq 0 \text{ for } \sigma \in \Sigma \text{ and } \alpha \in \mathcal{A}_\sigma \quad (4c)$$

227 where the $\alpha(j)$ are positive scalars that sum to one and $c(\sigma, \alpha)$ is the cost of taking action α in
228 state σ .

229 The reader will certainly notice similarities between the deterministic formulation (2) and the
230 stochastic formulation (4). Indeed, in both cases, the feasible vectors x have an interpretation as
231 “occupation measures” that provide the (expected) number of times a given state-action pair will
232 be visited.

233 An important difference is that the deterministic problem (2) corresponds to a network flow
234 problem (as seen in Example 1 and made formal in Section 3.1) while the stochastic formulation (4)
235 does not. This was described in detail in [16]. Because of stochasticity, arcs may have multiple head
236 nodes, corresponding to the set of states that could be reached under different random realizations
237 of the transition. Such an object is called a hypernetwork. This distinction of a network problem
238 versus a hypernetwork problem is important, since the methods of this paper extend dual-based
239 methods to solve network flow problems. To our knowledge, these methods do not have ready
240 analogies in the hypernetwork setting.

241 Another important difference is that, when the state and action sets are finite, the resulting
242 linear program (4) in this case is finite dimensional. Notice that the “infinite horizon” part of the
243 problem set up does not translate into an infinite formulation. The stationarity of the problem
244 allows us to finitely handle infinite time, and the LP formulation (4) is in the dimension of the
245 size of the state and action sets. Indeed, for the formulation of a stationary DP to be infinite
246 dimensional, either an infinite state space or infinite action space is needed (for [12, 13, 24] for
247 examples of this).

248 Our problem is infinite-dimensional because of nonstationarity. Although the state and action
249 sets are finite, data are indexed by time. Indeed, if one were to formulate our problem as a
250 nonhomogenous DP (like in (4)) would require an infinite state space Σ . This formulation is as
251 follows: state set

$$252 \quad \Sigma = \{(t, s) : t = 1, 2, \dots, s \in S_t\} \quad (5)$$

254 action sets $A_{(t,s)} = A_s$ for all t , costs $c((t, s), a) = c_t(s, a)$ for all states (t, s) and actions $a \in A_{(t,s)}$,

¹Puterman calls this the “dual” linear program of the MDP. In our context, it is more fitting to call it the “primal” linear program.

255 and transition probabilities $p_t((t', s')|(t, s), a) = 1$ if $s' = \tau_t(s, a)$ and $t' = t + 1$ and 0 otherwise.
 256 Under this approach, the state set Σ is infinite and unbounded and the formulation (4) is a CILP.
 257 As discussed in the introduction, existing papers in the literature on homogenous stochastic DPs
 258 do not apply to this X since those papers assume the state space Σ is compact. Clearly, the state
 259 space Σ in (5) is not compact.

260 3 Pure-supply infinite network flow problems

261 We now discuss a generalization of the network flow problem (2), which was introduced in [30]. We
 262 recall the necessary notions from that paper and describe how (2) is a special case of that setting.

263 Let $G = (\mathcal{N}, \mathcal{A})$ be a directed graph with countably many nodes $\mathcal{N} = \{1, 2, \dots\}$ and arcs
 264 $\mathcal{A} \subseteq \mathcal{N} \times \mathcal{N}$. Let $I(i)$ denote the set of nodes that are tails of arcs entering node i : $I(i) := \{j \in \mathcal{N} :$
 265 $(j, i) \in \mathcal{A}\}$. Similarly, the set of nodes that are heads of arcs leaving i is $O(i) := \{j \in \mathcal{N} : (i, j) \in \mathcal{A}\}$.
 266 Each arc (i, j) has *cost* c_{ij} . Each node has supply b_i . The countably-infinite network flow problem
 267 (CINF) problem on network $(\mathcal{N}, \mathcal{A}, b, c)$ is to find a real nonnegative flow vector x that minimizes
 268 the cost and maintains flow balance at every node:

$$269 \quad Z^* := \inf_x \sum_{(i,j) \in \mathcal{A}} c_{ij} x_{ij} \quad (6a)$$

$$270 \quad \text{(CINF)} \quad \text{s.t.} \quad \sum_{j \in O(i)} x_{ij} - \sum_{j \in I(i)} x_{ji} = b_i \text{ for } i \in \mathcal{N} \quad (6b)$$

$$271 \quad x_{ij} \geq 0 \text{ for } (i, j) \in \mathcal{A}. \quad (6c)$$

273 Let $Z(x) := \sum_{(i,j) \in \mathcal{A}} c_{ij} x_{ij}$. A feasible solution x to (CINF) is called a feasible flow.

274 For an arc $(i, j) \in \mathcal{A}$, node i is called the *tail node* of arc (i, j) and node j its *head node*. The
 275 *in-degree* and *out-degree* of node i in G are the cardinalities of $I(i)$ and $O(i)$, respectively. A graph
 276 is *locally finite* if every node has finite in- and out-degree. A *finite (undirected) path* in G is a finite
 277 sequence of distinct nodes i_1, i_2, \dots, i_n , where $(i_k, i_{k+1}) \in \mathcal{A}$ or $(i_{k+1}, i_k) \in \mathcal{A}$ for $k = 1, \dots, n - 1$.
 278 A *path to infinity* is a sequence of distinct nodes i_1, i_2, \dots where $(i_k, i_{k+1}) \in \mathcal{A}$ or $(i_{k+1}, i_k) \in \mathcal{A}$
 279 for $k = 1, 2, \dots$. We typically use P_{ij} to denote a finite path from node i to node j , and $P_{i\infty}$ to
 280 denote a path from node i to infinity. Two nodes i and j are *finitely connected* in G if there exists
 281 a finite path P_{ij} . Two nodes i and j are *connected at infinity* if G contains two paths to infinity,
 282 $P_{i\infty}$ and $P_{j\infty}$, that share no common nodes. Nodes i and j are *connected* if they are either finitely
 283 connected or connected at infinity. The graph G is *finitely connected* if all nodes i and j in G are
 284 finitely connected.

285 A *finite cycle* in G is a finite sequence of nodes $i_1, i_2, \dots, i_n, i_1$ where i_1, i_2, \dots, i_n is a path and
 286 either $(i_1, i_n) \in \mathcal{A}$ or $(i_n, i_1) \in \mathcal{A}$. An *infinite cycle*, also called a *cycle at infinity*, consists of two
 287 paths to infinity from some node i , (i, i_1, i_2, \dots) and (i, j_1, j_2, \dots) , where all intermediate nodes i_k

288 and j_ℓ are distinct.

289 We also need directed versions of these definitions. A *finite directed path* from i_1 to i_n , denoted
 290 $P_{i_1 i_n}^{\rightarrow}$, is a finite path i_1, \dots, i_n where $(i_k, i_{k+1}) \in A$ for all $k = 1, 2, \dots, n - 1$. We call $P_{i_1 i_n}^{\rightarrow}$ an
 291 *out-path* from i_1 and an *in-path* into node i_n . A *directed path from node i to infinity*, denoted
 292 $P_{i\infty}^{\rightarrow}$, is a path to infinity $P_{i\infty}$ where each arc in the path is directed *away* from node i . A *directed*
 293 *path from infinity to node i* , denoted $P_{i\infty}^{\leftarrow}$, is a path to infinity $P_{i\infty}$ where each arc in the path is
 294 directed *towards* node i . A *directed finite cycle* is a finite cycle that consists of a finite directed
 295 path i_1, \dots, i_n and the arc $(i_n, i_1) \in A$. A *directed cycle at infinity* is a cycle at infinity where both
 296 paths to infinity from a given node i are directed, one from infinity to i , and the other from i to
 297 infinity.

298 A graph is *acyclic* if it contains no finite or infinite directed cycles. A subgraph of G is a *forest*
 299 if it contains no (undirected) cycles. A connected forest is called a *tree*. A tree that contains all
 300 the nodes of G is called a *spanning tree*. We are a bit sloppy when it comes to subgraphs of G , and
 301 will think of them alternatively as sets of arcs, sets of nodes, or entire subgraphs, as the context
 302 requires.

303 An *in-tree rooted at node r* is a tree where from every node i in the tree there is a directed path
 304 to node r where all arcs are directed to node r . An *out-tree rooted at node r* has directed paths
 305 from node r to every other node in the tree, with now each arc directed away from node r . When
 306 r is designated as the node at infinity, we call the tree an *in-tree rooted at infinity* and has the
 307 property that every node i in the tree has a unique infinite directed path $P_{i\infty}^{\rightarrow}$.

308 Following [30], we will assume that the network $(\mathcal{N}, \mathcal{A}, b, c)$

309 (A1) is locally finite,

310 (A2) is finitely connected,

311 (A3) contains no finite or infinite directed cycles,

312 (A4) has finitely many nodes with in-degree 0

313 (A5) has integer-valued supplies; that is, b_i is integer for all $i \in \mathcal{N}$

314 (A6) has nonnegative supplies; that is, $b_i \geq 0$ for all $i \in \mathcal{N}$,

315 (A7) has uniformly-bounded supplies; that is, $b \in \ell_\infty(\mathcal{N})$ and so there exists a $\bar{b} = \|b\|_\infty$ is
 316 the uniform upper bound on all node supplies, and

317 (A8) the arc costs are nonnegative; that is, $c_{ij} \geq 0$ for all $(i, j) \in \mathcal{A}$.

318 Assumption (A8) is not assumed in [30] but is needed here for reasons that will be remarked on
 319 below.

320 A vector x satisfying constraints (6b) is a *basic flow* if the arcs $\{(i, j) \in \mathcal{A} : x_{ij} \neq 0\}$ form a
 321 forest in G . Theorem 3.13 of [28] shows that every forest can be extended to a spanning tree of
 322 G . This ensures that every basic flow can be associated with (at least one) spanning tree. A basic
 323 flow is a *basic feasible flow (bff)* if the flow is also nonnegative.

324 Until now we have not set any assumptions on the costs of arcs. To do so, we introduce another

325 concept: *stages* of nodes. As carefully detailed in [30], this requires some preprocessing. Identify all
 326 transshipment nodes with out-degree zero. Since no feasible flow will send positive flow along arcs
 327 into such nodes, they can be removed along with all of their incident arcs without loss of generality.
 328 Apply this rule recursively until no such nodes remain. Moreover, without loss of *feasibility*, each
 329 supply node has out-degree at least one — otherwise flow balance is violated. This establishes the
 330 following.

331 **Proposition 2** (Proposition 2.1 in [30]). In every feasible instance of (CINF) whose underlying
 332 instance satisfies (A1)–(A7), each node has out-degree at least one without loss of generality.

333 We can now define stages of nodes. Stage 0 is the set of all nodes with in-degree 0. From
 334 assumption (A4), this set is finite. Stage 1 consists of all nodes with in-degree 0 in the modified
 335 graph that results from removing all stage 0 nodes and their incident arcs. Observe that all Stage 1
 336 nodes are incident to Stage 0 nodes in the graph. Repeat this procedure to construct the remaining
 337 stages. Since the graph is acyclic, each node is contained in exactly one stage. Let \mathcal{S}_t denote the
 338 set of nodes in Stage t and $s(i)$ denote the stage of node i . By construction, each stage is a finite
 339 collection of nodes. A key property for our dual-based methods is the following.

340 **Lemma 1.** For every arc $(i, j) \in \mathcal{A}$, $s(i) < s(j)$.

341 *Proof.* Only after removing node i will it be possible that j has an in-degree of 0 since the arc (i, j)
 342 itself gives node j an in-degree of at least one. \square

343 Since there are countably-many nodes, they can be numbered $1, 2, \dots$. There are many possible
 344 numberings, but we require that the numberings of nodes obeys the staging. That is, for every
 345 $i \in \mathcal{S}_m$ and $j \in \mathcal{S}_n$ with $m < n$, node i is numbered before node j . Succinctly we write this as
 346 $i < j$.

347 We make the following assumptions on stages and structure of the costs of the graph.

348 (A9) there exist $\beta \in (0, 1)$ and $\gamma \in (0, +\infty)$ such that for every $(i, j) \in \mathcal{A}$, $|c_{ij}| \leq \gamma\beta^{s(i)}$,
 349 where β can be interpreted as a discount factor,

350 (A10) there exists a finite uniform upper bound G and the size of cardinality of each stage
 351 t ; that is $\#(\mathcal{S}_t) \leq G$ for all t .

352 It is straightforward to see (by the dominated convergence theorem) that these assumptions
 353 imply that the sequence c of costs is absolutely summable; that is, $\|c\|_1 < \infty$.

354 **Remark 2.** The uniform upper bound on the cardinality of each stage G is needed to get an
 355 analytical finite bound on the computational burden of each iteration of the algorithms we present
 356 below. Please see the proof of **Proposition 6** for details. Finiteness of each iteration can still be
 357 established as long as

$$358 \quad \sum \beta^t \#(\mathcal{S}_t) < \infty \tag{7}$$

359 holds, but an analytical bound is challenging to establish without additional structure. For sake of
 360 clarity, development for the more general condition (7) is not explored. Assumption (S2) is natural
 361 in the deterministic DP setting, since often the set of states S is not changing with time whereas
 362 costs and transitions between states do change.

363 **Definition 1.** Following [30], we call a network that satisfies assumptions (A1)–(A10) a *pure*
 364 *supply network*. A (CINF) problem with an underlying pure supply network is called a *pure supply*
 365 *problem*.

366 The name “pure supply” is inspired by assumption (A6), but also includes all the additional
 367 assumptions stated. As noted in [30], we can also handle the complementary setting where assump-
 368 tion (A6) is replaced by $b_i \leq 0$ for all $i \in \mathcal{N}$, by simply reversing arc directions and changing the
 369 signs of the demands.

370 3.1 Dynamic programming is a pure supply problem

371 We next show that deterministic dynamic programming falls within our focus class of network flow
 372 problems.

373 **Proposition 3.** The network flow problem (2) associated with dynamic program (1) is a pure-
 374 supply problem.

375 *Proof.* Let N denote the network underlying problem (2) and G its associated graph. Hence, G is
 376 locally finite (condition (A1)) under the stated assumption that $A_{s,t}$ is finite for all s and t . As for
 377 local connectedness (condition (A2)), this holds given our structure of action sets and transitions.
 378 Indeed, we have assumed that $A_{s,t} \neq \emptyset$ for all s, t and for every $s' \in S_{t+1}$ there exists an $s \in S_t$
 379 and $a \in A_{s,t}$ such that $s' = \tau_t(s, a)$. This ensures finite connectedness since all nodes can trace a
 380 path back to $(s_0, 0)$ and so are connected (in the undirected sense that is required). Turning to the
 381 nonexistence of directed cycles (condition (A3)), there exists no arc from a node (s, t) to a node
 382 (s', t') with $t' \leq t$. This makes both finite and infinite directed cycles impossible. Regarding 0
 383 in-degree (condition (A4)), the only node with in-degree 0 is the starting node $(s_0, 0)$ and so there
 384 are clearly finitely-many such nodes. Conditions (A5)–(A7) hold since the right-hand of (2) only
 385 takes on values 0 and 1. Turning to the stage structure, observe that stage $\mathcal{S}_t = \{(s, t) : s \in S_t\}$.
 386 Conditions (A9) and (A10) then follow immediately by boundedness of immediate costs (property
 387 (S1)), cost discounting by δ , and assumptions on the growth of states (property (S2)) stated in
 388 Section 2. Condition (A8) holds since $c_t(s, a)$ are nonnegative for all s, a , and t . \square

389 **Remark 3.** The previous result shows that the dynamic programming network flow problem
 390 can be studied using the methodology of [30]. As we argued in Lemma 14, the primal *shadow*
 391 *simplex method* of [17] also applies. One may also check that the primal simplex method of [34]
 392 is valid for this setting. The network structure of (2), if we explicitly add the implied constraints

393 $x_{(s,t)(s',t+1)} \leq 1$ to the formulation, does satisfy the basic assumptions of [34] as well as the critical
 394 condition that the flow between stages is uniformly bounded (Proposition 2.5). The latter property
 395 is needed to ensure their simplex method converges in optimal value.

396 On the other hand, primal simplex methods described in [16, 24] do not directly apply. Their
 397 methods crucially rely on nondegeneracy properties of the underlying hypernetwork that corre-
 398 sponds to a stochastic dynamic program. By contrast, (2) is highly degenerate since all but one
 399 node has a supply of zero. \triangleleft

400 3.2 Constructing trees and basic feasible flows in pure supply networks

401 The following properties of pure supply networks are used in our analysis of the dual-ascent method.
 402 Consider the following procedure:

403 **Procedure 1** (Constructing a spanning tree). Given a pure supply network,

- 404 (i) for every node i select a single outgoing arc a_i (such an arc is guaranteed to exist by
 405 **Proposition 2**), and
- 406 (ii) construct the subgraph T with arc set $\{a_i : i \in \mathcal{N}\}$.

407 **Lemma 2** (Lemma 4.2 in [30]). A subgraph T of a pure supply network is a spanning in-tree
 408 rooted at infinity if and only if it can be constructed by **Procedure 1**.

409 Consider also the following related procedure.

410 **Procedure 2** (Constructing a basic flow from a tree). Given a spanning tree T of a pure supply
 411 network, construct a flow x^T as follows:

- 412 (i) initially set $x_i^S = 0$ for all $i \in \mathcal{N}$
- 413 (ii) for each $i \in \mathcal{N}$, identify the unique path $P_{i\infty}$ from i to infinity in T , with forward arcs
 414 $P_{i\infty}^F$ and backward arcs $P_{i\infty}^B$, and add a flow of b_i to all arcs in $P_{i\infty}^F$ and remove a flow
 415 of b_i from all arcs in $P_{i\infty}^B$.

416 For a general spanning tree T , x^T need not be a basic *feasible* flow. However, when T is an
 417 in-tree rooted at infinity, this is indeed the case.

418 **Lemma 3** (Lemma 4.4 in [30]). If T is a spanning in-tree rooted at infinity in a pure supply
 419 network (e.g., if T is constructed by **Procedure 1**) then x^T is a basic feasible flow.

420 4 A dual-ascent method for pure-supply problems

421 We construct a simple dual-ascent method to solve (CINF), inspired by the success of dual-ascent
 422 methods for finite network flow problems. From [30], the dual problem to (6) is

$$423 \quad D^* := \max_{\pi} \sum_{i \in \mathcal{N}} b_i \pi_i \quad (8a)$$

424 (CINF D) s.t. $\pi_i - \pi_j \leq c_{ij}$ for $(i, j) \in \mathcal{A}$ (8b)

425 $\pi \in c_0,$ (8c)
426

427 where c_0 is the vector space of all sequences that converge to 0. Let $D(\pi) := \sum_{i \in \mathcal{N}} b_i \pi_i$.

428 Weak duality for (6) and (8) was proved in [30], which is critical to our DUAL-ASCENT METHOD.
429 We restate the result here for completeness.

430 **Theorem 1** (Weak duality, Theorem 7.3 of [30]). In an instance of (6) where assumption (A1)–
431 (A10) hold, every primal feasible x and dual feasible π satisfy $Z(x) \geq D(\pi)$. In particular, if there
432 exist a primal feasible x^* and dual feasible π^* such that $Z(x^*) = D(\pi^*)$ then x^* is an optimal
433 solution to (6) and π^* is an optimal solution to (8).

434 We also point out that there is a natural pairing between the space c_0 and the finite signed
435 measures over the time-indexed state and action pairs (this was discussed in Section 2.1). Thus,
436 the standard arguments of [2] for weak duality also hold in view of this pairing.

437 The idea of a dual-ascent method is to iteratively generate dual feasible solutions π^n that strictly
438 improves $D(\pi^n)$ (unless a termination condition is reached). Each iteration of our algorithm adjusts
439 π at a single node i , incrementing it by the slack value $s_{ij} = c_{ij} + \pi_j - \pi_i$ of constraint (8b) for
440 some j where $(i, j) \in \mathcal{A}$. This construction implies that π_i maintains the cost of a finite path from
441 node i to some node j in the graph. Our algorithm chooses the node i that admits the largest
442 possible increase in the value of π . That is, node i is chosen so that s_{ij} is as large as possible and
443 yet still maintains dual feasibility. Since the s_{ij} are nonnegative for all $(i, j) \in \mathcal{A}$, this implies that
444 π is strictly increased at each iteration unless all s_{ij} are zero. Careful details follow.

445 The following provides an alternate interpretation of the updating rule (12).

446 **Lemma 4.** For all i and n , the iterate π_i^n of the DUAL-ASCENT METHOD satisfies

447
$$\pi_i^n = \begin{cases} \pi_i^{n-1} + s_{i^n j^n}^n & \text{if } i = i^n \\ \pi_i^{n-1} & \text{otherwise.} \end{cases} \quad (13)$$

448 *Proof.* Observe that $c_{i^n j^n} + \pi_{j^n}^{n-1} = \pi_{i^n}^{n-1} + s_{i^n j^n}^n$ by (9) and so we can alternately express the
449 updating (12) of π_i^n as in (13). \square

450 **Remark 4.** It is important to point out the possibility that the same node i^n may be visited more
451 than once by the algorithm. Observe that when an arc $a = (i^{n_1}, j^{n_1})$ is selected at time n_1 , then
452 (12) ensures that $s_{i^{n_1} j^{n_1}}^{n_1+1} = 0$ when the algorithm next visits (9). That is,

453
$$\pi_{i^{n_1}}^{n_1} - \pi_{j^{n_1}}^{n_1} = c_{i^{n_1} j^{n_1}} \quad (14)$$

454 This implies $s_{i^{n_1} j^{n_1}}^{n_1+1} = 0$ and so node i^{n_1} will not be immediately selected as i^{n_1+1} at iteration
455 $n_1 + 1$. However, we now show it is possible for $s_{i^{n_1} j^{n_1}}^{n_2} > 0$ at a later iteration for some iteration

DUAL-ASCENT METHOD

1. (Initialization) Input a pure supply network (that is, an infinite network satisfying assumptions (A1)–(A10)) and set $\pi_i^0 = 0$ for all $i \in \mathcal{N}$ and $n = 1$.
2. (Construct slacks) Set

$$s_{ij}^n \leftarrow c_{ij} + \pi_j^{n-1} - \pi_i^{n-1} \quad \text{for all } (i, j) \in \mathcal{A}, \quad (9)$$

$$s_i^n \leftarrow \min_{j \in O(i)} s_{ij}^n \quad \text{for all } i \in \mathcal{N} \quad (10)$$

$$s^n \leftarrow \max_{i \in \mathcal{N}} s_i^n. \quad (11)$$

3. (Check for termination) If $s^n = 0$ then *terminate*. Else go to 4.
4. (Update π^n) Set $i^n \in \arg \max_{i \in \mathcal{N}} s_i^n$ and $j^n \in \arg \min_{j \in O(i)} s_{ij}^n$ and update

$$\pi_i^n \leftarrow \begin{cases} \pi_{j^n}^{n-1} + c_{i^n j^n} & \text{if } i = i^n \\ \pi_i^{n-1} & \text{otherwise.} \end{cases} \quad (12)$$

5. (Update n) Increment n to $n + 1$ and go to step 2.

456 n_2 , allowing the possibility that $i^{n_2} = i^{n_1}$. Suppose $i^{n_2-1} = j^{n_1}$ (that is, the head node of arc a
 457 is selected immediately before iteration n_2) and neither node incident to arc a was selected since
 458 iteration n_1 . This implies that (14) still holds at iteration $n_2 - 1$; that is,

$$459 \quad \pi_{i^{n_1}}^{n_2-1} - \pi_{j^{n_1}}^{n_2-1} = c_{i^{n_1} j^{n_1}}. \quad (15)$$

460 Now, suppose arc (j^{n_1}, j^{n_2-1}) is the arc selected in iteration $n_2 - 1$. This implies

$$461 \quad \pi_{j^{n_1}}^{n_2} = \pi_{j^{n_1}}^{n_2-1} + c_{j^{n_1} j^{n_2-1}} \quad (16)$$

462 and

$$463 \quad \pi_{i^{n_1}}^{n_2} = \pi_{i^{n_1}}^{n_2-1}. \quad (17)$$

464 Putting equations (15)–(17) together implies

$$\begin{aligned} 465 \quad \pi_{i^{n_1}}^{n_2} - \pi_{j^{n_1}}^{n_2} &= \pi_{i^{n_1}}^{n_2-1} - \pi_{j^{n_1}}^{n_2-1} - c_{i^{n_1} j^{n_1}} \\ 466 &= c_{i^{n_1} j^{n_1}} - c_{i^{n_1} j^{n_1}} \\ 467 &< c_{i^{n_1} j^{n_1}}, \end{aligned}$$

469 under the assumption of nonnegative costs and assuming for sake of argument that $c_{j^{n_1} j^{n_2-1}} > 0$.
 470 This implies that $s_{i^{n_1} j^{n_1}}^{n_2} > 0$, allowing for the possibility that $i^{n_2} = i^{n_1}$. This return to a previously
 471 processed node may cause the reader to worry the algorithm cycles. The issue of cycling is addressed

472 below in [Remark 6](#). \triangleleft

473 The following observations give us greater insights into the DUAL-ASCENT METHOD. The first
474 key observation is that iterates are dual feasible solutions.

475 **Proposition 4** (Dual feasibility of iterates). For all n , the iterates π^n of the DUAL-ASCENT
476 METHOD are feasible to (CINFD).

477 *Proof.* We argue inductively. For the base ($n = 0$) case, plugging π^0 into (8b) reveals $0 \leq c_{ij}$ for
478 all $(i, j) \in \mathcal{A}$ and clearly $\pi^0 \in c_0$. Thus, π^0 satisfies (8b) and (8c) and so is dual feasible.

479 For the inductive step, suppose π^{n-1} is dual feasible and argue the same is true for π^n . It
480 suffices to check that $s_{ij}^{n+1} \geq 0$ for all arcs $(i, j) \in \mathcal{A}$. If arc (i, j) is such that neither i nor j are
481 equal to i^n then $s_{ij}^{n+1} \geq 0$ follows by induction. Next suppose $j = i^n$ in arc $(i, j) \in \mathcal{A}$ and hence

$$\begin{aligned}
 482 \quad s_{ii^n}^{n+1} &= c_{ii^n} + \pi_{i^n}^n - \pi_i^n \\
 483 \quad &= c_{ii^n} + \pi_{i^n}^n - \pi_i^{n-1} \\
 484 \quad &\geq c_{ii^n} + \pi_{i^n}^{n-1} - \pi_i^{n-1} \\
 485 \quad &= s_{ii^n}^n \\
 486 \quad &\geq 0, \\
 487
 \end{aligned}$$

488 where the second equality holds since $\pi^n = \pi^{n-1}$, the first inequality uses the fact that $\pi_{i^n}^n \geq \pi_{i^n}^{n-1}$
489 from (13) and the inductive hypothesis that $s_{i^n j^n}^n \geq 0$, and the third equality holds by the definition
490 of $s_{ii^n}^n$ and the final inequality holds by the inductive hypothesis.

491 Next, suppose (i, j) is such that $i = i^n$. Then we have for all $j \in O(i)$,

$$\begin{aligned}
 492 \quad s_{i^n j}^{n+1} &= c_{i^n j} + \pi_j^n - \pi_{i^n}^n \\
 493 \quad &= c_{i^n j} + \pi_j^n - \pi_{i^n}^{n-1} - s_{i^n j^n}^n \\
 494 \quad &= s_{i^n j}^n - s_{i^n j^n}^n \\
 495 \quad &\geq 0, \\
 496
 \end{aligned}$$

497 where the second equality uses (13), the third equality uses the definition of $s_{i^n j}^n$ and the inequality
498 uses the fact that $s_{i^n j^n} = \min_{j \in O(i^n)} s_{i^n j}^n$.

499 It only remains to argue that $\pi^n \in c_0$. Note that the updating process of π^{n-1} in Step 3 of the
500 DUAL-ASCENT METHOD changes at most one entry from π^{n-1} to π^n . Hence, since $\pi_i^0 = 0$ for all
501 $i \in \mathcal{N}$, π^n has at most n nonzero entries, implying $\pi^n \in c_0$. \square

502 **Proposition 4** implies, in particular, that $s_{ij}^n \geq 0$ for all $(i, j) \in \mathcal{A}$ and for all n . The next result
503 discusses the termination condition, which is related to the values of the slacks.

504 **Proposition 5** (Optimality upon termination). If the termination condition $s^n = 0$ in the DUAL-
505 ASCENT METHOD is reached at iteration n , π^{n-1} is an optimal dual solution. Moreover, a primal
506 optimal flow can be constructed at termination.

507 *Proof.* By **Proposition 4**, π^{n-1} is a dual feasible solution. If $s^n = 0$ then $s_i^n = 0$ for all i and thus for
508 every node i there exists an arc (i, j) with $\pi_i^{n-1} - \pi_j^{n-1} = c_{ij}$. Construct a tree T using **Procedure 1**
509 where (i, j) is chosen for each node i . By **Lemma 2**, the resulting tree T is a spanning in-tree rooted
510 at infinity and so the flow x^T constructed using **Procedure 2** is a basic feasible flow by **Lemma 3**.

511 Next, we argue that π^{n-1} and x^T have the same objective values; that is, $D(\pi^{n-1}) = Z(x^T)$.
512 Indeed,

$$\begin{aligned}
513 \quad Z(x^T) &= \sum_{(i,j) \in \mathcal{A}} c_{ij} x_{ij}^T \\
514 &= \sum_{(i,j) \in T} c_{ij} x_{ij}^T \\
515 &= \sum_{i \in \mathcal{N}} \sum_{(j,k) \in P_{i\infty}} b_i c_{jk} \\
516 &= \sum_{i \in \mathcal{N}} b_i \left[\sum_{(i,j) \in P_{i\infty}} (\pi_j^{n-1} - \pi_k^{n-1}) \right] \\
517 &= \sum_{i \in \mathcal{N}} b_i \pi_i^{n-1} \\
518 &= D(\pi^{n-1}), \\
519
\end{aligned}$$

520 where the first equality is the definition of $Z(x^T)$, the second equality uses the fact that $x_{ij}^T = 0$ for
521 $(i, j) \notin T$, the third equality is the method of constructing basic feasible solutions in **Procedure 2**,
522 the fourth equality uses the fact that $s_{ij} = 0$ for all arcs (i, j) in T and so $c_{ij} = \pi_i^{n-1} - \pi_j^{n-1}$, the
523 fifth equality comes from telescoping along the path $P_{i\infty}$, and the final equality is the definition of
524 $D(\pi^{n-1})$.

525 Since $Z(x^T) = D(\pi^{n-1})$ and x^T is primal feasible and π^{n-1} is dual feasible, **Theorem 1** implies
526 π^{n-1} is an optimal dual solution and x^T is a primal optimal flow. \square

527 The next results discuss interpretations of the dual iterates π^n .

528 **Lemma 5.** For all i and n , the iterate π_i^n of the DUAL-ASCENT METHOD is the cost of a finite
529 out-tree from node i (and possibly an empty tree with no arcs and cost 0).

530 *Proof.* This follows by induction. For the base case, note that $\pi_i^0 = 0$ is the cost of an empty
531 directed tree. The inductive hypothesis is that for all i , π_i^{n-1} is the cost of a finite out-tree from
532 node i . By the updating formula (12), when $i \neq i^n$, $\pi_i^n = \pi_i^{n-1}$ and clearly π_i^{n-1} is the cost of a
533 finite out-tree from node i by the inductive hypothesis. When $i = i^n$, we know that there is a finite

534 out-tree T from node j^n with cost $\pi_{j^n}^{n-1}$. Consider adding arc (i^n, j^n) to T . The only worry is that
535 adding this arc creates a cycle, which only happens where j^n is in T already. However, if j^n is in T
536 already, adding (i^n, j^n) creates a finite directed cycle. Under assumption (A3) this cannot happen.
537 Hence, adding (i^n, j^n) to T creates a finite out-tree from node i and thus π_i^n is the cost of that
538 newly created tree. This completes the argument. \square

539 **Remark 5.** Note that it need not be the case that the T defined in the proof of the previous lemma
540 is a directed path. Indeed, the possibility that a node i is visited more than once by the algorithm
541 (as discussed in Remark 4) gives rise to this possibility. \triangleleft

542 The next results concern the “computational finiteness” of the algorithm. In particular, there
543 is an important step (equation (11)) that naïvely could involve infinite work if the maximum that
544 defines s^n is attained at all. The next two results resolve this issue. Observe that the minimum
545 defining s_i^n in (10) is attained since $O(i)$ is a finite set by assumption (A1).

546 **Lemma 6.** The slack variable s^n defined in (11) is well-defined; that is, despite the fact \mathcal{N} is
547 infinite, the maximum defining s^n is finite and attained.

548 *Proof.* First, we show each s_{ij}^n is finite. All π_i can be bounded above by $\|c\|_1$ since π_i is the length of
549 a finite out-tree in the graph (by Lemma 5) which is bounded by the sum of all costs $\|c\|_1$ of all arcs
550 in the graph (this uses assumption (A8)). This implies that $s_{ij}^n = c_{ij} + \pi_j^{n-1} - \pi_i^{n-1} \leq 4\|c\|_1 < \infty$.
551 This gives a uniform upper bound on the s_{ij}^n , and so s_i^n and s^n are all finite.

552 Next, we claim that s_i^n converges to 0 as $i \rightarrow \infty$. This will imply there exists an M such that
553 $\max_{i \in \mathcal{N}} s_i^n \leq \max_{i \in \{1, 2, \dots, M\}} s_i^n$. By Lemma 1, the stage $s(j)$ of node j for any $(i, j) \in \mathcal{A}$ is in a
554 stage greater than $s(i)$. Hence, the out-trees that defines π_i^n and π_j^n consist of arcs with tail nodes
555 in stage at least $s(i)$. As i goes to infinity, $s(i) \rightarrow \infty$ since nodes are numbered according to stages
556 and each stage contains finitely many nodes. Thus, $\pi_i^n \rightarrow 0$, $\pi_j^n \rightarrow 0$, and $c_{ij} \rightarrow 0$ as $i \rightarrow \infty$.
557 This implies $s_{ij}^n \rightarrow 0$ as $i \rightarrow \infty$ for all $j \in O(i)$ and thus $s_i^n \rightarrow 0$ as $i \rightarrow \infty$. This completes the
558 argument. \square

559 **Proposition 6** (Finite implementability). If $s^n \neq 0$ then steps 2 to 5 of the DUAL-ASCENT
560 METHOD can be executed in finite time at iteration n of the algorithm.

561 *Proof.* It suffices to argue that, for a given n , only finitely many arcs $(i, j) \in \mathcal{A}$ need to be visited in
562 step 2 to compute s^n (in equation (11)). This also means that i^n and j^n in Step 4 can be computed
563 in finite time and equation (12) can be executed in finite time.

564 The way we argue that only finitely many $(i, j) \in \mathcal{A}$ need to be visited is as follows. Since
565 $s^n \neq 0$ then there exists a first node i^* such that $s_{i^*}^n > 0$. Let $\delta_n \triangleq s_{i^*}^n > 0$. We will show how to
566 construct an M_n such that

$$567 \quad s_j^n \leq \delta_n \text{ for all } j \geq M_n. \quad (18)$$

568

569 This means that the $\arg \max_{i \in \mathcal{N}} s_i^n$ in equation (11) contain an element less than M_n . This implies
 570 that only the arcs (i, j) incident with a node $i, j \leq M_n$ need to be processed in equation (9) and
 571 only nodes $i \leq M_n$ need to be processed in equation (10).

572 Thus, the goal turns to constructing an M_n . We start by developing s_j^n from the right-hand
 573 side of (18):

$$\begin{aligned}
 574 \quad s_j^n &= \min_{k \in O(j)} s_{jk}^n \\
 575 \quad &= \min_{k \in O(j)} (c_{jk} + \pi_k^{n-1} - \pi_j^{n-1}) \tag{19} \\
 576
 \end{aligned}$$

577 using equations (9) and (10). Towards (18), we want to upper bound on (19). For the first term in
 578 the minimization we have

$$579 \quad c_{jk} \leq \gamma \beta^{s(j)} \tag{20}$$

580 from (A9). Next, we bound π_{n-1}^k , the second term in (19). From Lemma 5, π_k^n is the cost of a
 581 (finite) out-tree from node k . Thus, all the nodes in the tree associated with node k must be stage
 582 $s(k)$ or later (and thus certainly the nodes involved in the arcs of that tree are labeled k or higher).
 583 Accordingly, we can write:

$$\begin{aligned}
 584 \quad \pi_k^n &\leq \sum_{(p,q): p \geq k} c_{pq} \\
 585 \quad &\leq \left(\sum_{s=s(k)}^{\infty} \sum_{(p,q): s(p)=s} \gamma \beta^s \right) \\
 586 \quad &= \gamma G \sum_{s=s(k)}^{\infty} \beta^s \tag{21} \\
 587
 \end{aligned}$$

588 Hence, we can continue from (19):

$$\begin{aligned}
 589 \quad s_j^n &= \min_{k \in O(j)} s_{jk}^n \\
 590 \quad &\leq \min_{k \in O(j)} (c_{jk} + \pi_k^{n-1}) \tag{22}
 \end{aligned}$$

$$591 \quad \leq \min_{k \in O(j)} (\gamma \beta^{s(k)} + \gamma G \sum_{s=s(k)}^{\infty} \beta^s) \tag{23}$$

$$592 \quad \leq \gamma \beta^{s(j)} + \gamma G \sum_{s=j}^{\infty} \beta^s g(s) \tag{24}$$

$$593 \quad \leq \gamma \left(\beta^{s(j)} + G \beta^j \frac{1}{1-\beta} \right)$$

594
$$\leq \gamma \left(\beta^{s(j)} + G \beta^{s(j)} \frac{1}{1-\beta} \right) \tag{25}$$

595
$$= \gamma \beta^{s(j)} \left(1 + \frac{G}{1-\beta} \right) \tag{26}$$

596

597 where (22) follows by dropping the negative π_j^{n-1} terms and (23) uses (20) and (21). As for (24),
 598 this follows since $\beta^{s(j)} \leq \beta^{s(k)}$ for all $k \in O(j)$ by Lemma 1. Finally, (25) holds since $s(j) \leq j$ (each
 599 stage has at least one node).

600 Recall our target condition (18). In (26) we have

601
$$s_j^n \leq \gamma \beta^{s(j)} \left(1 + \frac{G}{1-\beta} \right)$$

602

603 and so we can guarantee (18) for any j that satisfies

604
$$\beta^{s(j)} \left(1 + \frac{G}{1-\beta} \right) \leq \frac{\delta_n}{\gamma}$$

605

606 We want to turn this into an expression of the form $j \geq \dots$. Observe that

607
$$\beta^{s(j)} \left(1 + \frac{G}{1-\beta} \right) \leq \frac{\delta_n}{\gamma}$$

608
$$\iff \beta^{s(j)} \leq \frac{\frac{\delta_n}{\gamma}}{\left(1 + \frac{G}{1-\beta} \right)}$$

609
$$\iff s(j) \geq \frac{\ln \left(\frac{\frac{\delta_n}{\gamma}}{\left(1 + \frac{G}{1-\beta} \right)} \right)}{\ln \beta}.$$

610

611 That is, as long as $j \geq M_n$ where

612
$$M_n \triangleq s^{-1} \left(\left\lceil \frac{\ln \left(\frac{\frac{\delta_n}{\gamma}}{\left(1 + \frac{G}{1-\beta} \right)} \right)}{\ln \beta} \right\rceil \right) \tag{27}$$

613

614 then our target condition (18) holds. □

615 An important observation is that the termination condition $s^n = 0$ of the DUAL-ASCENT
 616 METHOD cannot, in general, be verified in finite time. Interestingly, if the algorithm reaches a
 617 stage where the condition is true, then the current solution is optimal (via Proposition 5) but there
 618 is no way (in finite time) to know that this is the case. However, Proposition 6 says that if we ignore
 619 Step 3 and we happen to be in the condition that $s^n \neq 0$, then all other steps of the algorithm

620 can be processed in finite time. It is in this sense that we say that the DUAL-ASCENT METHOD is
 621 finitely-implementable. A similar phenomenon was observed in [16].

622 We now turn to the asymptotic performance of the DUAL-ASCENT METHOD. That is, we
 623 consider the limiting behavior of the iterates as n tends towards infinity.

624 **Lemma 7** (Convergence to dual feasibility). The sequence of iterates π^n of the DUAL-ASCENT
 625 METHOD converges pointwise to a dual feasible vector π^* as $i \rightarrow \infty$. That is, $\pi_i^n \rightarrow \pi_i^*$ as $n \rightarrow \infty$
 626 for all $i \in \mathcal{N}$.

627 *Proof.* We first show that the π^n converge pointwise. As argued in the proof of **Lemma 6**, the π_i^n
 628 are nonnegative and uniformly bounded above by $\|c\|_1$. Moreover, since $c_{ij} \geq 0$, π_i^n is monotone
 629 nondecreasing as a sequence in n . Hence, for each i , the sequence π_i^n converges to some limit, say
 630 π_i^* . Let $\pi^* = (\pi_i^* : i = 1, 2, \dots)$.

631 Next, we argue that π^* is dual feasible. From **Proposition 4** we know $c_{ij} + \pi_j^n - \pi_i^n \geq 0$ for all
 632 n . Hence $\lim_{n \rightarrow \infty} (c_{ij} + \pi_j^n - \pi_i^n) \geq 0$ or $c_{ij} + \pi_j^* - \pi_i^* \geq 0$ and π^* satisfies (8b). It remains to argue
 633 that π^* is in c_0 . Suppose otherwise. There exists a subsequence $\{i_k\}_{k=1}^\infty$ of nodes such that

$$634 \quad \pi_{i_k}^* > \epsilon \text{ for some } \epsilon > 0 \tag{28}$$

635 As argued in the previous paragraph, $\pi_{i_k}^n \rightarrow \pi_{i_k}^*$ as $n \rightarrow \infty$. Via **Lemma 5**, we know $\pi_{i_k}^n$ is the
 636 cost of a finite out-tree T^n from node i_k . By **Lemma 1**, all arcs in T^n have tails in stage $s(i_k)$ or
 637 higher. Thus by assumption (A9), the cost of every arc in T^n is bounded by $\gamma\beta^{s(i)}$. Moreover, we
 638 can upper bound the number of arcs in T^n by $\sum_{t=s(i_k)}^{s_n} G$, where s_n is the stage of the largest head
 639 node of an arc of T^n (such an arc exists since T^n is finite). Thus, $\pi_{i_k}^n \leq \sum_{t=s(i_k)}^{s_n} G\gamma\beta^{s(i_k)}$, which
 640 converges to 0 as k goes to infinity. This implies that there exist k_0 and n_0 such that $\pi_{i_k}^n \leq \epsilon$ for
 641 $n \geq n_0$ and $k \geq k_0$. Since $\pi_{i_k}^n \rightarrow \pi_{i_k}^*$ as $n \rightarrow \infty$, this implies that $\pi_{i_k}^* \leq \epsilon$ for k sufficiently large.
 642 This contradicts (28), completing the proof. \square

643 **Lemma 8** (Slacks tend to zero). The s_i^n defined in (10) of the DUAL-ASCENT METHOD satisfy
 644 $\lim_{n \rightarrow \infty} s_i^n \rightarrow 0$ for all $i \in \mathcal{N}$.

645 *Proof.* We proceed by contradiction. Suppose not so that there exists an $i \in \mathcal{N}$, an $\epsilon > 0$, and a
 646 subsequence $s_i^{n_k}$ of the s_i^n such that $s_i^{n_k} > \epsilon > 0$ for all k . This implies that $s^n > \epsilon$ for infinitely
 647 many n . Hence, from (13) in **Lemma 4**, an entry of π^n is increased by at least $\epsilon > 0$ infinitely often.
 648 But this is impossible since π_i^n is the cost of a finite out-tree from node i , which is bounded above
 649 by $\|c\|_1$. \square

650 **Theorem 2** (Convergence to dual optimality). The sequence of iterates π^n generated by the
 651 DUAL-ASCENT METHOD converge pointwise to an optimal dual solution π^* .

652 *Proof.* By [Lemma 7](#), the limit sequence π^* exists and is dual feasible. Let $s_{ij}^* := c_{ij} + \pi_j^* - \pi_i^*$,
653 which denotes the slacks of the dual constraints (8b) for the dual solution π^* . Since $\pi_j^n \rightarrow \pi_j^*$ and
654 $\pi_i^n \rightarrow \pi_i^*$ as $n \rightarrow \infty$,

$$655 \quad s_{ij}^n \rightarrow c_{ij} + \pi_j^* - \pi_i^* = s_{ij}^* \quad (29)$$

656 as $n \rightarrow \infty$. From [Lemma 8](#), for all i , $s_i^n \rightarrow 0$. By the pigeon-hole principle, for every i there exists
657 a $j \in O(i)$ such that $s_i^n = s_{ij}^n$ for infinitely many n . Restricting to the subsequence n_k of the n
658 such that $s_i^{n_k} = s_{ij}^{n_k}$, $s_{ij}^* = \lim_k s_i^{n_k} = 0$, using (29) and [Lemma 8](#), respectively (and noting all
659 subsequences of convergent sequences have the same limit). This implies that for each node i , we
660 can identify an arc (i, j) such that $s_{ij}^* = 0$.

661 By [Lemma 2](#), the subgraph T^* of G consisting of the identified arcs where $s_{ij}^* = 0$ form a
662 spanning in-tree rooted at infinity. Thus, by [Lemma 3](#), the associated basic feasible flow x^{T^*} ,
663 constructed by [Procedure 2](#), is a basic feasible flow. We note that $Z(x^{T^*}) = D(\pi^*)$. This is
664 precisely the same argument captured in the string of equalities in the proof of [Proposition 5](#),
665 replacing T with T^* and π^{n-1} with π^* . Finally, by [Theorem 1](#), $Z(x^{T^*}) = D(\pi^*)$ implies π^* is an
666 optimal dual solution (and x^{T^*} is a primal optimal flow). \square

667 **Remark 6.** It is useful to note that it is impossible for the sequence of iterates to cycle; that is,
668 revisit an earlier value for π . The reason is that the updating of π (as reformulated in (13)) always
669 occurs when s_{i^n, j^n}^n is strictly bigger than zero (otherwise the method terminates in step 3). \triangleleft

670 **Remark 7.** The proof of [Theorem 2](#) establishes the existence of a primal optimal solution x^{T^*} and
671 a dual optimal solution π^* with the same objective value, that is, $Z(x^{T^*}) = D(\pi^*)$. In other words,
672 the DUAL-ASCENT METHOD can be used to establish the strong duality of (CINF) and (CINF D)
673 under assumptions (A1)–(A10). This was established in [30] using a primal simplex method, and
674 in previous papers (such as [26, 34]) using topological arguments. \triangleleft

675 As the above results show, the DUAL-ASCENT METHOD produces a sequence of improving dual
676 feasible solutions that pointwise converge to an optimal dual solution. In other words, the DUAL-
677 ASCENT METHOD is an approach to solve the *dual* of the pure supply problem (6). Another usage
678 is to develop optimality error bounds for the primal problem in finite time. In [30], the authors
679 propose a primal simplex method that generates an improving sequence of primal feasible iterates
680 that converge in value to optimality. Running the primal simplex method and dual ascent method
681 in parallel produces a primal feasible solution x^n and dual feasible solution π^n after n iterations of
682 each algorithm. The optimality error $Z(x^n) - Z^*$ of the primal is thus bounded by $D(\pi^n) - Z(x^n)$
683 after finitely many iterations of both algorithms. In the next section, we show how to adjust the
684 DUAL-ASCENT METHOD to produce a sequence of primal and dual feasible iterates simultaneously,
685 and whose structures are related, by devising a primal-simplex method.

5 A primal-dual method for pure supply problems

We now develop a primal-dual method. It is standard practice in the finite-dimensional setting to develop first a dual-ascent method and then modify it slightly (mostly by just tracking more of what is already generated in the dual ascent case) to construct a primal-dual method that maintains a sequence of both primal and dual feasible iterates. The connection between the primal and dual feasible solution is through the following notion.

Definition 2. An arc (i, j) is balanced with respect to $\pi \in \mathbb{R}^N$ if $\pi_i - \pi_j = c_{ij}$.

In the finite-dimensional setting, the dual feasible iterates π^n are built so that they correspond to costs of a forest of balanced arcs (with respect to π^n). This forest can then be extended to a spanning tree, and thus a primal feasible flow (typically using some max flow algorithm). As the algorithm proceeds, arcs are added to this forest and eventually there is a dual feasible iterate found that corresponds (in cost) to a spanning tree of balanced arcs. Then an argument very similar to the proof of [Proposition 5](#) produces an optimal dual solution and optimal primal feasible solution.

The issue in our setting is that the dual iterates in the DUAL-ASCENT METHOD do not correspond to forests of balanced arcs. This was noted in [Remark 4](#), that arcs can become “unbalanced” as the method proceeds. This highlights a difference between our DUAL-ASCENT METHOD and the typical methods in the finite-dimensional setting. We carefully leverage the pure supply structure of the underlying network to avoid running max-flow calculations in each iteration (as, for instance, is done in Chapter 7 of [\[7\]](#)). We adapt the method to maintain as much of its underlying simplicity as possible.

Accordingly, we adapt the method not to maintain a forest of balanced arcs (as in the finite-dimensional case), but instead of growing set of balanced *nodes*, defined as follows.

Definition 3. A node i is balanced with respect to $\pi \in \mathbb{R}^N$ if there exists a $j \in O(i)$ such that (i, j) is balanced with respect to π . That is, a node is balanced if it has an outgoing arc that is balanced.

The updating step in the DUAL-ASCENT METHOD has the potential to unbalance arcs (and thus possibly unbalance nodes). So in our revised algorithm (called the PRIMAL-DUAL METHOD) we add a “rebalancing step” that balances (potentially different) arcs to assure that every previously balanced node remains balanced. Accordingly, the set of balanced nodes grows monotonically in size with every iterations of the algorithm, while the set of balanced arcs typically loses arcs and gains others as the algorithm proceeds.

In the limit, we will show that every node becomes balanced, and so a similar conclusion to the finite case can be drawn. Namely, there is a limiting dual solution that corresponds to a limiting primal feasible solution where every arc in the underlying spanning tree is balanced.

We now look more carefully at the two major differences between the DUAL-ASCENT METHOD and the PRIMAL-DUAL METHOD: Step 5 (the rebalancing step) and Step 6 (the step that will

PRIMAL-DUAL METHOD

1. (Initialization) Input a pure supply network (that is, an infinite network satisfying assumptions (A1)–(A10)) and set $\pi_i^0 = 0$ for all $i \in \mathcal{N}$, $B^0 = \{i \in \mathcal{N} : c_{ij} = 0 \text{ for some } j \in O(i)\}$, and $n = 1$. For every $i \in \mathcal{N}$ select a single arc $a_i = (i, j)$ such that $c_{ij} = \min\{c_{ij} : j \in O(i)\}$ and $T^0 = \{a_i : i \in \mathcal{N}\}$
2. (Construct initial slacks) Set

$$s_{ij}^n \leftarrow c_{ij} + \pi_j^{n-1} - \pi_i^{n-1} \quad \text{for all } (i, j) \in \mathcal{A}, \quad (30)$$

$$s_i^n \leftarrow \min_{j \in O(i)} s_{ij}^n \quad \text{for all } i \in \mathcal{N} \quad (31)$$

3. (Termination check) Set $s^n \leftarrow \max_{i \in \mathcal{N}} s_i^n$. If $s^n = 0$ then *terminate*. Else go to Step 4.
4. (Balancing step) Set $i^n \in \arg \max_{i \in \mathcal{N}} s_i^n$, $j^n \in \arg \min_{j \in O(i)} s_{ij}^n$, and

$$\pi_i^n \leftarrow \begin{cases} \pi_{j^n}^{n-1} + c_{i j^n} & \text{if } i = i^n \\ \pi_i^{n-1} & \text{otherwise.} \end{cases} \quad (32)$$

$$s_{ij}^n \leftarrow c_{ij} + \pi_j^n - \pi_i^n \text{ for all } (i, j) \in \mathcal{A} \text{ with } i = i' \text{ or } j = i' \quad (33)$$

$$s_i^n \leftarrow \min_{j \in O(i)} s_{ij}^n \text{ for all } i \in \mathcal{N} \quad (34)$$

$$B^n \leftarrow B^{n-1} \cup \{i^n\} \quad (35)$$

$$T^n \leftarrow T^{n-1} \cup \{(i^n, j^n)\} \setminus \{(i, j) \in T^{n-1} : i = i^n\} \quad (36)$$

5. (Rebalancing step)

while $\{i \in B^n | s_i^n > 0\} \neq \emptyset$

$$i' \leftarrow \max\{i \in B^n | s_i^n > 0\} \quad (37)$$

$$j' \leftarrow \text{a node } j' \in O(i') \text{ such that } s_{i'j'}^n = s_{i'}^n \quad (38)$$

$$\pi_{i'}^n \leftarrow \pi_{j'}^n + c_{i'j'} \quad (39)$$

$$s_{ij}^n \leftarrow c_{ij} + \pi_j^n - \pi_i^n \text{ for all } (i, j) \in \mathcal{A} \text{ with } i = i' \text{ or } j = i' \quad (40)$$

$$s_i^n \leftarrow \min_{j \in O(i)} s_{ij}^n \text{ for all } i \in \mathcal{N} \quad (41)$$

$$T^n \leftarrow T^n \cup \{(i', j')\} \setminus \{(i, j) \in T^n : i = i'\} \quad (42)$$

endwhile

6. (Update x^n) Set

$$x^n \leftarrow x^{T^n} \quad (43)$$

where x^{T^n} is as constructed in **Procedure 2**.

7. (Update n) Set $n \leftarrow n + 1$ and go to Step 3.

722 generate primal-feasible iterates). The rebalancing step is essential as it allows us to grow the set
 723 of balanced nodes. Before establishing this, we first point out that the algorithm does not get
 724 caught in the while loop in Step 5.

725 **Lemma 9.** In every iteration of the PRIMAL-DUAL METHOD, the while loop in Step 5 finitely
 726 terminates.

727 *Proof.* It suffices to prove that the set $\{i \in B^n : s_i^n > 0\}$ eventually becomes empty. Suppose i' and
 728 j' are chosen as in (37) and (38) for one iteration of the while. Setting $s_{i'j'}^n$ to 0 in step (40) removed
 729 node i' from the set $\{i \in B^n : s_i^n > 0\}$ at the end of the iteration of the while. Next, we
 730 argue that this node i' never returns to the set $\{i \in B^n : s_i^n > 0\}$.

731 The only way that i' re-enters the set $\{i \in B^n : s_i^n > 0\}$ is if $\pi_{j'}^m$ is changed from its value $\pi_{j'}^n$ at
 732 some later iteration m . However, by definition of i' , we know $s_j^n = 0$ for all $j \in B^n$ with $j > i$ (by
 733 the definition of i' as the maximum element of the set $\{i \in B^n : s_i^n > 0\}$). This implies that $\pi_{j'}$ will
 734 not be adjusted in the course of the rebalancing step, since only the tail nodes of arcs processed by
 735 the step have their π value adjusted, and all arcs (j', k) with j' as their tail node have $s_{j'k}^n = 0$ and
 736 this slack will not be adjusted since the same property holds for the nodes k .

737 Hence, a node is removed from $\{i \in B^n : s_i^n > 0\}$ at every iteration of the while, and since B^n
 738 is initially a finite set and no nodes are added to B^n in the rebalancing step, this implies that
 739 $\{i \in B^n : s_i^n > 0\}$ is eventually empty, breaking the while loop. \square

740 **Remark 8.** Implicit in the argument above is that the only nodes that be rebalanced are prede-
 741 cessor nodes of the node i^n that is balanced in Step 4 of the PRIMAL-DUAL METHOD. How such
 742 nodes can become unbalanced was explored in concrete terms in Remark 4. \triangleleft

743 As the algorithm proceeds, n is successively iterated, and according to (3), the set B^n grows
 744 with every iteration. We now show that the set B^n contains only balanced nodes. This explains
 745 the naming of Steps 4 and 5 as balancing and rebalancing steps, respectively.

746 **Proposition 7.** Every node in B^n after the rebalancing step (Step 4 of the Primal-Dual Method)
 747 is balanced with respect to π^n .

748 *Proof.* The proof is by induction. This property is true for B^0 by construction since every node i
 749 in B^0 has an arc $j \in O(i)$ such that $c_{ij} = 0$ and so

$$750 \quad \pi_i^0 - \pi_j^0 = c_{ij}$$

752 since both sides are equal to 0. For the inductive step, note that in (35), the node i^n that is added
 753 to B^n is balanced by the nature of the updating in (32). Indeed, arc (i^n, j^n) is balanced in step
 754 (32) (and, in particular $s_{i^n j^n}^n$ is set to 0 in (33)) and so i^n is balanced when added to B^n in (35).

755 It only remains to argue that the nodes in B^n added at an earlier stage remain balanced with
756 respect to the new value π^n . For a node $i \in B^n$, if s_i^n is unchanged in (34), then it remains at
757 $s_i^n = 0$ and so node i remains balanced. If, however, $s_i^n > 0$ after step (34), this is addressed in
758 the rebalancing step. Once this node is processed as i' in (37) then after step (41) we have $s_i^n = 0$
759 and so node i is again balanced. As argued in Lemma 9, the while loop terminates with $s_i^n = 0$ for
760 all $i \in B^n$. That is, all nodes in B^n remain balanced with respect to π^n at end of the rebalancing
761 step. \square

762 Next, we show that Step 6 generates a sequence of primal feasible flows.

763 **Proposition 8.** For all n , the iterates x^n of the PRIMAL-DUAL METHOD are feasible to (CINF).

764 *Proof.* We first show that T^n contains exactly one outgoing arc for every node $i \in \mathcal{N}$. We establish
765 this by induction. The base case for T^0 is true by construction. For the inductive hypothesis, note
766 that T^n is updated in (36) and revised in (42). By the inductive hypothesis, T^{n-1} has a single
767 outgoing arc for every node $i \in \mathcal{N}$ and so in (36) the set $\{(i, j) \in T^{n-1} : i = i^n\}$ is a singleton.
768 Thus, the update takes out one outgoing arc (i^n, j) of i^n from T^{n-1} and replaces it with another
769 outgoing arc (i^n, j^n) of i^n . Hence, T^n maintains a single outgoing arc for every node after (36).
770 By the same logic, this property is maintained every time (42) is invoked in the rebalancing step.
771 Thus, in (43), T^n always contains a single outgoing arc for every node, and hence x^n is a basic
772 feasible flow by Lemmas 2 and 3. \square

773 The proof of dual feasibility mimics that of the DUAL-ASCENT METHOD, applying the logic of
774 the balancing step to the rebalancing step. Details are omitted.

775 **Proposition 9** (cf. Proposition 4). For all n , the iterates π^n of the PRIMAL-DUAL METHOD are
776 feasible to (CINF).

777 The previous results explain some of the logic of the PRIMAL-DUAL METHOD. It works to build
778 balanced nodes and generates a sequence of both primal and dual feasible solutions. The concepts
779 come together in establishing a bound on the gap in value between the primal and dual iterates.

780 **Theorem 3.** For every iteration n , $Z(x^n) - D(\pi^n) \leq \sum_{T^n \setminus T_B^n} c_{ij} x_{ij}^n$, where T_B^n denotes the arcs in
781 T^n that are balanced with respect to π^n .

782 *Proof.* This proof uses the following notation. As x^n is constructed by Procedure 2 we may define
783 $P_{i\infty}$ as the unique directed path to infinity from node i in the spanning tree T^n . Using an analog
784 of Lemma 5, which applies equally to the π^n generated by the PRIMAL-DUAL METHOD, let T_i^n
785 denote the finite out-tree of node i whose cost is represented by π_i^n . Note that the set of arcs in
786 T_i^n must contain the arcs P_i^n in $P_{i\infty}$ with tail nodes in B^n . This is because every arc added to T^n
787 in (36) and (42) that is on a directed path from node i is captured in T_i^n .

788 Using this notation we observe that (with careful justification below):

$$789 \quad Z(x^n) - D(\pi^n) = \sum_{i \in \mathcal{N}} b_i c(P_{i\infty}) - \sum_{i \in \mathcal{N}} b_i \pi_i^n \quad (44)$$

$$790 \quad = \sum_{i \in \mathcal{N}} b_i (c(P_{i\infty}) - \pi_i^n) \quad (45)$$

$$791 \quad = \sum_{i \in \mathcal{N}} b_i (c(P_{i\infty}) - c(T_i^n)) \quad (46)$$

$$792 \quad \leq \sum_{i \in \mathcal{N}} b_i (c(P_{i\infty}) - c(P_i^n)) \quad (47)$$

$$793 \quad = \sum_{i \in \mathcal{N}} b_i c(P_{i\infty} \setminus P_i^n) \quad (48)$$

$$794 \quad = \sum_{(i,j) \in T^n \setminus T_B^n} c_{ij} x_{ij}^n. \quad (49)$$

796 Step (44) uses [Procedure 2](#). Step (46) uses the definition of T_i^n . Since the set of arcs in T_i^n contain
797 the arcs P_i^n in $P_{i\infty}$ with tail nodes in B^n , this justifies dropping the costs of arcs in T_i^n not in
798 P_i^n . The direction of the inequality follows since the costs of all arcs are nonnegative. Step (49)
799 uses the fact that P_i^n is a subset of $P_{i\infty}$ for all i . Finally, step (49) uses the understanding from
800 [Procedure 2](#) about how x^n is constructed by sending flow b_i along the unique paths $P_{i\infty}$ from node
801 i to infinity, for each arc i . This uses the fact that, since there are no demand nodes, *all* flow b_i
802 originating from node i must traverse the arcs in $P_{i\infty} \setminus P_i^n$ (which are precisely the arcs $T^n \setminus T_B^n$ in
803 the index of the sum in (49)). \square

804 An immediate consequence of [Propositions 8](#) and [9](#) and [Theorem 3](#) and weak duality ([Theorem 1](#))
805 is the following non-asymptotic error bounds on the quality of the primal and dual feasible
806 solutions generated by the PRIMAL-DUAL METHOD.

807 **Corollary 1.** For every iteration n , $Z(x^n) - Z^* \leq \sum_{T^n \setminus T_B^n} c_{ij} x_{ij}^n$ and $D^* - D(\pi^n) \leq \sum_{T^n \setminus T_B^n} c_{ij} x_{ij}^n$.

808 This is a non-asymptotic bound on the performance of both the primal method and dual as-
809 cent method. Moreover, [Theorem 3](#) yields asymptotic convergence results for the PRIMAL-DUAL
810 METHOD. Observe that the optimality error bound value $\sum_{T^n \setminus T_B^n} c_{ij} x_{ij}^n$ depends on how many
811 nodes are balanced, and thus how many arcs are in $T^n \setminus T_B^n$. The next two results show that all
812 nodes will eventually be balanced.

813 **Lemma 10** (Slacks tend to zero, cf. [Lemma 8](#)). The s_i^n updated in (34) and (41) of the PRIMAL-
814 DUAL METHOD satisfy $\lim_{n \rightarrow \infty} s_i^n \rightarrow 0$ for all $i \in \mathcal{N}$.

815 The proof of this lemma is analogous to that of [Lemma 8](#), adapted to the primal-dual setting,
816 and thus omitted.

817 **Proposition 10.** All nodes are eventually balanced. That is, for all $i \in \mathcal{N}$ there exists an n_i such
818 that node i is balanced with respect to π^n for all $n \geq n_i$.

819 *Proof.* Suppose, by way of contradiction, that node i is never balanced and thus $i \notin B_n$ for all
820 $n \geq 0$. This implies, in particular since $i \notin B_0$, that

$$821 \quad \min_{j \in O(i)} c_{ij} > 0. \quad (50)$$

823 Also, for all n and $k \in O(i)$ we have

$$824 \quad s_{ik}^n = c_{jk} + \pi_j^n - \pi_i^n \geq c_{ij} \quad (51)$$

825 since i is not in B_n and $\pi_j^n \geq 0$ for all $j \in O(i)$. Thus, from (50) and (51) we have

$$826 \quad \min_{k \in O(i)} s_{ij}^n \geq \min_{j \in O(i)} c_{ij} > 0,$$

828 and so $\lim_{n \rightarrow \infty} s_i^n > 0$. This contradicts [Lemma 10](#), thus completing the proof. \square

829 **Lemma 11.** The optimality error bound $\sum_{T^n \setminus T_B^n} c_{ij} x_{ij}^n$ from [Theorem 3](#) (and [Corollary 1](#)) con-
830 verges to 0 as $n \rightarrow \infty$.

831 *Proof.* Since, by [Proposition 10](#), all nodes are eventually balanced, there exists an n_s such that all
832 nodes in the first $s - 1$ stages are in B_{n_s} . This implies that $T^n \setminus T_B^n$ can only contain arcs with tail
833 nodes in stage s or later for $n \geq n_s$. Thus, the cost of every arc in $T^n \setminus T_B^n$ can be bounded by $\gamma\beta^s$.
834 That is,

$$835 \quad \sum_{T^n \setminus T_B^n} c_{ij} x_{ij}^n \leq \sum_{k=s}^{\infty} \sum_{i \in S_k} \sum_{j \in O(i)} c_{ij} x_{ij}^n$$

$$836 \quad \leq \|b\|_{\infty} \gamma G \sum_{k=s}^{\infty} \beta^k \quad (52)$$

838 for $n \geq n_s$, where the second inequality uses assumptions (A7)–(A10). Taking n sufficiently large,
839 we can send $s \rightarrow \infty$ which sends the right-hand side (52) to zero under assumption (A10). This
840 gives the result. \square

841 **Theorem 4.** The iterates π^n and x^n converge in value to D^* and Z^* . That is, $D(\pi^n) \rightarrow D^*$ and
842 $Z(x^n) \rightarrow Z^*$.

843 *Proof.* This is immediate from [Corollary 1](#) and [Lemma 11](#). \square

844 In [Theorem 2](#), the stronger result of the convergence of π^n to an optimal solution π^* , and
845 not just convergence in value, was established. On the primal side, we cannot guarantee solution

846 convergence. The reason is that there is a choice of arcs in the spanning tree sets T^n , which may
 847 not converge to a unique spanning tree in the limit.

848 However, we can also guarantee finite implementability of the PRIMAL-DUAL METHOD, as was
 849 argued in [Proposition 6](#) for the DUAL-ASCENT METHOD. The same result holds for the PRIMAL-
 850 DUAL METHOD. Here both the primal and dual solutions need only to be constructed on a finite
 851 subgraph corresponding to nodes in B^n , since as [Lemma 11](#) showed, costs of arcs outside of T_B^n can
 852 be made negligible. It should be noted that we cannot guarantee *finite convergence* of the overall
 853 algorithm, the termination condition in step 3 will, in general, be unreachable in finite time.

854 5.1 Application to dynamic programming

855 Returning to the deterministic DP problem, as established in [Proposition 3](#), it corresponds to a pure
 856 supply network flow problem and so the DUAL-ASCENT METHOD and PRIMAL-DUAL METHOD
 857 both apply. To make this concrete, we may write the dual of (2) as

$$858 \quad \max_{\pi} \pi_{(s_0,0)} \tag{53a}$$

$$859 \quad \text{s.t. } \pi_{(s,t)} - \pi_{(s',t+1)} \leq \delta^t c_{(s,t)(s',t+1)} \quad \text{for all } t \geq 0 \text{ and } ((s,t), (s',t+1)) \in \mathcal{A} \tag{53b}$$

$$860 \quad \pi \in c_0. \tag{53c}$$

862 **Remark 9.** As initially discussed in [Section 2.1](#), there is a connection between the dual of (2)
 863 given above in (53) and the dual formulation of the stochastic DP in (4):

$$864 \quad \max_v v(s_0) \tag{54a}$$

$$865 \quad \text{s.t. } v(s) - \delta \sum_{j \in S} p(j|s,a)v(j) \leq c(s,a) \quad \text{for all } s \in S \text{ and } a \in A_s \tag{54b}$$

867 where $v(s)$ has the interpretation of the cost-to-go of state s .

868 These two duals ((53) and (54)) also have related interpretations. The optimal solution $\pi_{s,t}^*$ to
 869 (53) has the interpretation of being the cost of an infinite path from state (s,t) to infinity. Thus,
 870 the optimal choice of $\pi_{s,t}^*$ corresponds to the cost of an optimal policy originating in state s at time
 871 t . Similarly, $v(s)$ has the interpretation of being the value of an optimal policy leaving state s .
 872 These two interpretations are analogous.

873 To our knowledge, the DUAL-ASCENT METHOD and PRIMAL-DUAL METHOD are the first
 874 dual-based algorithms proposed to solve (2) and (53). These dual based methods directly provide
 875 computable bounds on the value error from optimal for the strategy provided by a separate primal
 876 algorithm or within the primal-dual algorithm for the n th iterate. These bounds would, in general,
 877 be superior to the crude bound heretofore provided by the maximum discounted cost-to-go as, for
 878 example, in traditional finite horizon approximations.

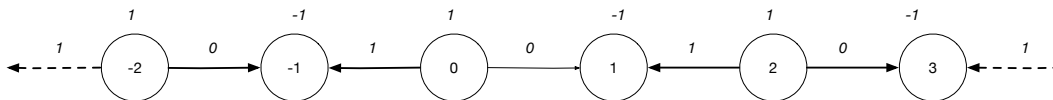


Figure 2: Illustration of a network with supply and demand nodes where a sequence of spanning trees fails to converge to a spanning tree.

6 Conclusion

In this paper, we establish a dual-ascent and primal-dual method for deterministic DP. In fact, we develop the methodology in the more general class of pure supply network flow problems. We show that these methods have desirable properties, including finite implementability, value and solution convergence (in the case of the dual-ascent method).

We should remark that the dual-ascent method studied here is a simple one, where at each iteration only a single arc is balanced (or relaxed). More general dual-ascent methods (such as those described in [7] and [6]) are considered in the finite network case. Iterations of our dual-ascent method most closely resembles a *single-node relaxation* (or coordinate ascent) approach proposed in Section 3.3 of [6]. Single-node relaxations have been shown to be computationally efficient in practice, but for general finite networks (such as those with both supply and demand nodes) such relaxations do not suffice to compute an optimal dual solution, they must be combined with more general relaxation steps. The reason we can confine ourselves to single node relaxations is the simple structure of the pure supply network.

For future work, one could explore analogs of more general dual-ascent methods from the finite-network settings and adapt them to the infinite setting. However, this path is fraught with potential difficulties, as demonstrated in the following example.

Example 2. Consider the network in Figure 2 with b_i indicated next to each i : odd-numbered nodes have supply 1 and even-numbered nodes have supply -1 , i.e., demand 1. Any spanning tree in this network consists of all arcs except for one. Bold arcs in the figure depict one such tree (arc $(0, 1)$ is excluded), and values next to the arcs form the corresponding basic flow. This is the unique feasible flow in this network; note that it is degenerate and a tree constructed by omitting the arc $(i, i + 1)$ for any even i has the same corresponding basic feasible flow.

Consider the sequence of spanning trees that exclude the arcs $(0, 1)$, $(2, 3)$, $(4, 5)$, etc. The limit of this sequence (in the product discrete topology defined in [30]) is the entire network, which is not a spanning tree.

One reason this example is problematic is as follows. In the pure supply setting, the trees T^n always correspond to spanning trees due to Lemma 3. Moreover, the limit tree T^* (studied in the proof of Theorem 2) is also a spanning tree because of Lemma 3. The above example shows that this need not be true for the mixed supply and demand case. The dual-ascent and primal-dual

909 methods in this paper strongly leverage the properties of the pure supply network (and particularly,
 910 **Lemma 3**) and so careful thought must be put into adapting to more general settings.

911 Acknowledgments

912 We would like to thank the associate editor and two referees for their many comments that improved
 913 the clarity of the paper. We also thank Marina A. Epelman for several enlightening discussions
 914 that helped improve the paper. The first author thanks the Chicago Booth of Business for its
 915 generous financial research support. The second author is supported in part by the National
 916 Science Foundation Grant CMMI-1333260.

917 A Proof of Proposition 1

918 We begin with some preliminary lemmas. To state the lemmas we need the notion of the *character-*
 919 *istic vector* $\chi^S \in \{0, 1\}^{\mathcal{A}}$ of a subset of arcs $S \subseteq \mathcal{A}$ that is defined by $\chi_{(s,t)(s',t+1)}^S = 1$ if $((s, t), (s', t+$
 920 $1)) \in S$ and 0 if $((s, t), (s', t + 1)) \in \mathcal{A} \setminus S$. Note that $Z(\chi^S) = \sum_{((s,t),(s',t+1)) \in S} \delta^t c_{(s,t)(s',t+1)}$ is the
 921 sum of the discounted costs of the arcs in S .

922 **Lemma 12.** The following hold:

- 923 (i) Every integral solution x to (2) is the characteristic vector of a path P_x from $(s_0, 0)$ to
 924 infinity and so $Z(x) = Z(\chi^{P_x})$.
- 925 (ii) Conversely, every path P from $(s_0, 0)$ to infinity corresponds to an integral feasible
 926 solution x_P to (2), where $Z(\chi^P) = Z(x_P)$.

927 *Proof.* To establish (i), note that (3b)–(3e) and integrality imply that the arcs with flow 1 for any
 928 integral solution x precisely form a path P from $(s_0, 0)$ to infinity. Thus, $x = \chi_P$. For (ii), observe
 929 that the characteristic vector of a path from $(s_0, 0)$ satisfies constraints (3b)–(3e). The result then
 930 follows. \square

931 **Lemma 13.** The following hold:

- 932 (i) Every policy d to (1) gives rise to a unique path P_d from $(s_0, 0)$ to infinity such that
 933 $V(\pi(d)_{s_0}) = Z(\chi^{P_d})$, where χ^{P_d} is the characteristic vector of a path P_d from $(s_0, 0)$ to
 934 infinity.
- 935 (ii) Conversely, every path P from $(s_0, 0)$ to infinity consisting of arcs $(s_0, 0), (s_1, 1), (s_2, 2), \dots$
 936 corresponds to a set of policies D_P where each policy d in that set satisfies:

$$937 \quad d(s, t) \begin{cases} = a \in A_t \text{ such that } \tau_t(s_t, a) = s_{t+1} & \text{if } s = s_t \\ \in A_t & \text{if } s \in S_t \text{ and } s \neq s_t \end{cases} \quad (55)$$

938 and $V(\pi(d), s_0) = Z(\chi^P)$ for every $d \in D_P$.

939 *Proof.* To establish (i) note that given a policy d and starting with initial state s_0 at time 0, the
 940 policy determines the unique path P_d consisting of arcs $(s_0, 0), (s_1, 1), (s_2, 2), \dots$ where $s_{t+1} =$
 941 $\tau_t(s_t, d(s_t, t))$ for $t = 0, 1, 2, \dots$. It is straightforward to see that $V(\pi(d), s_0) = Z(\chi^{P_d})$.

942 For (ii), clearly each d constructed in (55) has $s_{t+1} = \tau_t(s_t, d(s_t, t))$ for $t = 0, 1, 2, \dots$. This is
 943 all the information needed to specific the cost $V(\pi(d), s_0)$, as the choice of $d(s, t)$ for $s \in S_t$ with
 944 $s \neq s_t$ is irrelevant to the outcome of the dynamic program. Hence, $V(\pi(d), s_0) = Z(\chi^P)$ for every
 945 $d \in D_P$. \square

946 The following result is standard in finite-horizon versions of the problem. The argument here
 947 relies on a couple of useful results in the CILP literature.

948 **Lemma 14.** Problem (2) possesses an optimal flow that is integer-valued.

949 *Proof.* Recall Proposition 2.7 of [17]: a CILP has an optimal extreme point solution if

- 950 (a) the feasible region is non-empty,
- 951 (b) every constraint has a finite number of variables,
- 952 (c) the feasible region is bounded by some vector u ; that is, $x_{(s,t)(s',t+1)} \leq u_{(s,t)(s',t+1)}$ for
 953 all $((s, t), (s', t+1)) \in \mathcal{A}$ for as feasible x , and
- 954 (d) $\sum_{((s,t),(s',t+1)) \in \mathcal{A}} \delta^t c_{(s,t)(s',t+1)} u_{(s,t)(s',t+1)} < \infty$.

955 Condition (a) is clearly satisfied since every path from $(s_0, 0)$ to infinity corresponds to a feasible
 956 solution by Lemma 12(ii). Condition (b) follows by assumption (A1). As for (c), the vector
 957 $u_{(s,t)(s',t+1)} = 1$ for all $((s, t), (s', t+1)) \in \mathcal{A}$ suffices. Indeed, the only source of flow in the network
 958 is from node $(s_0, 0)$ with a supply of 1, hence no arc will ever have a flow larger than 1 (note that
 959 we do not impose this upper bound of flow explicitly, but it is implied by the data). Thus, (c)
 960 holds. As for (d) this follows since

$$961 \sum_{((s,t),(s',t+1)) \in \mathcal{A}} \delta^t c_{(s,t)(s',t+1)} u_{(s,t)(s',t+1)} \leq G \sum_{t=0}^{\infty} \delta^t \cdot \bar{c} \cdot 1 = G \bar{c} \frac{1}{1-\delta} < \infty,$$

962 where the inequality follows from (S1) and (S2). Thus, (2) has an optimal extreme point solution.
 963 Next, by Theorem 3.14 in [28], every extreme point solution is integral, since the right-hand sides
 964 of the constraints (3b)–(6b) are integral. Taken together, this implies (2) has an integer solution
 965 that is optimal. \square

966 *Proof of Proposition 1.* Lemmas 12 and 13 imply that (a) every feasible policy d to (1) gives rise
 967 to an integer flow x_d of (2) where $V(\pi(d), s_0) = Z(x_d)$ and (b) every integer flow x of (2) gives
 968 rise to policies d_x that obey (55) where P_x is the path associated with integer flow x described in
 969 Lemma 12(i) and $Z(x) = V(\pi(d_x), s_0)$ for any such policy.

970 We now establish (i). By (a), d^* has a corresponding integer flow x^* with $V(\pi(d^*), s_0) = Z(x^*)$.
 971 Next, by (b), every integer flow x of (2) gives rise to a policy d_x such that $V(\pi(d_x), s_0) = Z(x)$.
 972 Taken together this implies that

$$973 \quad Z(x^*) = V(\pi(d^*), s_0) \leq V(\pi(d_x), s_0) = Z(x),$$

974 where the inequality uses optimality of the policy d^* in (1). Thus, x^* has the minimum cost among
 975 all integer flows to (2). By Lemma 14, this implies that x^* is an optimal flow for (2). This completes
 976 (i).

977 Returning to (ii), there exists an optimal *integer flow* x^* to (2). Let d^* be any policy that obeys
 978 (55) where $P = P_{x^*}$ with $V(\pi(d^*), s_0) = Z(x^*)$. Every feasible policy d corresponds to an integer
 979 vector x_d with $V(\pi(d), s_0) = Z(x_d)$ and so

$$980 \quad V(\pi(d), s_0) = Z(x_d) \geq Z(x^*) = V(\pi(d^*), s_0),$$

981 where the inequality follows by the optimality of x^* in (2). Thus, d^* is an optimal policy since its
 982 value is no greater than that of any other feasible policy. \square

983 References

- 984 [1] Jeffrey M. Alden and Robert L. Smith. Rolling horizon procedures in nonhomogeneous Markov
 985 decision processes. *Operations Research*, 40(3-supplement-2):S183–S194, 1992. 2
- 986 [2] Edward J. Anderson and Peter Nash. *Linear Programming in Infinite-Dimensional Spaces:
 987 Theory and Applications*. Wiley, 1987. 2, 14
- 988 [3] James C. Bean and Robert L. Smith. Conditions for the existence of planning horizons.
 989 *Mathematics of Operations Research*, 9(3):391–401, 1984. 2
- 990 [4] James C Bean and Robert L Smith. Optimal capacity expansion over an infinite horizon.
 991 *Management Science*, 31(12):1523–1532, 1985. 1
- 992 [5] James C Bean, Jack R Lohmann, and Robert L Smith. Equipment replacement under tech-
 993 nological change. *Naval Research Logistics (NRL)*, 41(1):117–128, 1994. 1
- 994 [6] Dimitri P Bertsekas. *Linear Network Optimization: Algorithms and Codes*. MIT Press, 1991.
 995 30
- 996 [7] Dimistris Bertsimas and John N. Tsitsiklis. *Introduction to Linear Optimization*. Athena,
 997 1997. 3, 7, 23, 30

- 998 [8] Christian Bès and Suresh P. Sethi. Concepts of forecast and decision horizons: Applications
999 to dynamic stochastic optimization problems. *Mathematics of Operations Research*, 13(2):
1000 295–310, 1988. 2
- 1001 [9] Abraham Charnes, Jacques Dreze, and Merton Miller. Decision and horizon rules for stochastic
1002 planning problems: A linear example. *Econometrica*, pages 307–330, 1966. 2
- 1003 [10] Eric V. Denardo. *Dynamic Programming: Models and Applications*. Dover, 2003. 2
- 1004 [11] François Dufour and Tomas Prieto-Rumeau. Finite linear programming approximations of con-
1005 strained discounted Markov decision processes. *SIAM Journal on Control and Optimization*,
1006 51(2):1298–1324, 2013. 3
- 1007 [12] François Dufour and Tomas Prieto-Rumeau. Stochastic approximations of constrained dis-
1008 counted Markov decision processes. *Journal of Mathematical Analysis and Applications*, 413
1009 (2):856–879, 2014. 3, 4, 8
- 1010 [13] Peyman Mohajerin Esfahani, Tobias Sutter, Daniel Kuhn, and John Lygeros. From infinite
1011 to finite programs: Explicit error bounds with applications to approximate dynamic program-
1012 ming. *SIAM Journal on Optimization*, 28(3):1968–1998, 2018. 3, 4, 8
- 1013 [14] Eugene A. Feinberg and Adam Shwartz. *Handbook of Markov Decision Processes: Methods
1014 and Applications*. Kluwer, 2002. 2
- 1015 [15] Archis Ghatge. Circumventing the Slater conundrum in countably infinite linear programs.
1016 *European Journal of Operations Research*, 246(3):708–720, 2015. 3
- 1017 [16] Archis Ghatge and Robert L. Smith. A linear programming approach to non stationary infinite-
1018 horizon Markov decision processes. *Operations Research*, 61:413–425, 2013. 2, 8, 13, 21
- 1019 [17] Archis Ghatge, Dushyant Sharma, and Robert L. Smith. A shadow simplex method for infinite
1020 linear programs. *Operations Research*, 58(4):865–877, 2010. 2, 12, 32
- 1021 [18] Onesimo Hernández-Lerma and Jean B. Lasserre. *Discrete-time Markov Control Processes:
1022 Basic Optimality Criteria*, volume 30. Springer, 2012. 2
- 1023 [19] Wallace J. Hopp, James C. Bean, and Robert L. Smith. A new optimality criterion for non-
1024 homogeneous Markov decision processes. *Operations Research*, 35(6):875–883, 1987. 2
- 1025 [20] Angeliki Kamoutsis, Tobias Sutter, Peyman Mohajerin Esfahani, and John Lygeros. On infinite
1026 linear programming and the moment approach to deterministic infinite horizon discounted
1027 optimal control problems. *IEEE control systems letters*, 1(1):134–139, 2017. 6

- 1028 [21] David E Kaufman and Robert L Smith. Fastest paths in time-dependent networks for intelli-
1029 gent vehicle-highway systems application. *Journal of Intelligent Transportation Systems*, 1(1):
1030 1–11, 1993. 1
- 1031 [22] Diego Klabjan and Daniel Adelman. Existence of optimal policies for semi-Markov decision
1032 processes using duality for infinite linear programming. *SIAM Journal on Control and Opti-
1033 mization*, 2006. 2
- 1034 [23] Diego Klabjan and Daniel Adelman. An infinite-dimensional linear programming algorithm
1035 for deterministic semi-Markov decision processes on Borel spaces. *Mathematics of Operations
1036 Research*, 32(3):528–550, 2007. 2
- 1037 [24] Ilbin Lee, Marina A. Epelman, H. Edwin Romeijn, and Robert L. Smith. Simplex algorithm for
1038 countable-state discounted Markov decision processes. *Operations Research*, 65(4):1029–1042,
1039 2017. 2, 8, 13
- 1040 [25] Franco Modigliani and Franz E. Hohn. Production planning over time and the nature of the
1041 expectation and planning horizon. *Econometrica*, pages 46–66, 1955. 2
- 1042 [26] Sevnaz Nourollahi and Archis Gbate. Duality in convex minimum cost flow problems on infinite
1043 networks and hypernetworks. *Networks*, 70(2):98–115, 2017. 3, 22
- 1044 [27] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*.
1045 Wiley, 2014. 2, 7, 8
- 1046 [28] H. Edwin Romeijn, Dushyant Sharma, and Robert L. Smith. Extreme point characterizations
1047 for infinite network flow problems. *Networks*, 48(4):209–22, 2006. 10, 32
- 1048 [29] Sheldon M. Ross. *Stochastic Processes*. Wiley, 1996. 2
- 1049 [30] Christopher Thomas Ryan, Robert L Smith, and Marina A Epelman. A simplex method for
1050 uncapacitated pure-supply infinite network flow problems. *SIAM Journal on Optimization*, 28
1051 (3):2022–2048, 2018. 2, 3, 4, 9, 10, 11, 12, 13, 14, 22, 30
- 1052 [31] Sarah M. Ryan, James C. Bean, and Robert L. Smith. A tie-breaking rule for discrete infinite
1053 horizon optimization. *Operations Research*, 40(1-supplement-1):S117–S126, 1992. 2
- 1054 [32] Naci Saldi, Serdar Yüksel, and Tamás Linder. On the asymptotic optimality of finite approxi-
1055 mations to Markov decision processes with Borel spaces. *Mathematics of Operations Research*,
1056 42(4):945–978, 2017. 3, 4
- 1057 [33] Irwin E. Schochetman and Robert L. Smith. Convergence of selections with applications in
1058 optimization. *Journal of Mathematical Analysis and Applications*, 155(1):278–292, 1991. 2

- 1059 [34] Thomas C. Sharkey and H. Edwin Romeijn. A simplex algorithm for minimum-cost network-
1060 flow problems in infinite networks. *Networks*, 52(1):14–31, 2008. [2](#), [3](#), [12](#), [13](#), [22](#)
- 1061 [35] Robert L Smith and Rachel Q Zhang. Infinite horizon production planning in time-varying
1062 systems with convex production and inventory costs. *Management Science*, 44(9):1313–1320,
1063 1998. [1](#)